TUGAS AKHIR –SM141501

KLASIFIKASI JENIS KENDARAAN BERGERAK BERBASIS*GEOMETRIC INVARIANT MOMENT*

ZAMROJI HARIYANTO
NRP 1211 100 058

DosenPembimbing
Dr. Budi Setiyono, S.Si, MT

JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT -SM141501

MOVING VEHICLE CLASSIFICATION BASED ON GEOMETRIC INVARIANT MOMENT

ZAMROJI HARIYANTO
NRP 1211 100 058

Supervisor
Dr. Budi Setiyono, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Science
SepuluhNopember Institute of Technology
Surabaya 2015

LEMBAR PENGESAHAN

KLASIFIKASI JENIS KENDARAAN BERGERAK BERBASIS *GEOMETRIC INVARIANT MOMENT*

MOVING VEHICLE CLASSIFICATION BASED ON GEOMETRIC INVARIANT MOMENT

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk memperoleh Gelar Sarjana Sains
Pada Bidang Studi Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

ZAMROJI HARIYANTO
NRP. 1211 100 058

Menyetujui,
Dosen Pembimbing,

Dr. Budi Setiyono, S.Si, MT
NIP. 19720207 199702 1 001

Mengetahui,
Ketua Jurusan Matematika
FMIPA ITS

Prof. Dr. Erna Apriliani, M.Si
NIP. 19660414 199102 2 001

Surabaya, Januari 2015

KLASIFIKASI JENIS KENDARAAN BERGERAK BERBASIS *GEOMETRIC INVARIANT MOMENT*

Nama Mahasiswa : Zamroji Hariyanto
NRP : 1211 100 058
Jurusan : Matematika
DosenPembimbing : Dr. Budi Setiyono, S.Si, MT

Abstrak

Penelitian ini bertujuan untuk mengklasifikasikan dan menghitung jenis kendaraan bergerak di suatu jalan raya dengan menggunakan pengolahan citra digital. Objek dari penelitian ini adalah video rekaman arus kendaraan. Kemudian dilakukan pemilihan ROI (*Region of Interest*) yang digunakan sebagai area klasifikasi dan penghitungan kendaraan bergerak. Selanjutnya setiap ROI pada *frame* dari video tersebut diproses menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model* untuk mendapatkan citra *foreground* dan *background*. Citra *foreground* difilter, dihilangkan bayangannya, dan dilakukan proses morfologi untuk menghilangkan *noise* pada citra. Kemudian citra hasilnya dianalisis *blob* untuk mendapatkan bagian piksel yang merupakan objek bergerak. Tahap berikutnya, setiap objek yang diperoleh diklasifikasi dengan *Geometric Invariant Moment*. Setiap objek akan dihitung nilai momen invarian pertama (ϕ_1) dan lebarnya kemudian dijadikan sebagai parameter untuk klasifikasi kendaraan. Apabila nilai parameter memenuhi nilai interval maka objek akan dikenali jenisnya sesuai dengan jenis yang didefinisikan pada interval tersebut. Hasil pengujian perangkat lunak dapat mengklasifikasikan kendaraan dalam tiga jenis kendaraan yaitu motor, mobil, dan truk/bus dengan tingkat akurasi tertinggi 94,27%.

Kata Kunci: Penghitungan Kendaraan, Pengklasifikasian Kendaraan, Pengolahan citra digital, *Gaussian Mixture Model* (*GMM), Geometric Invariant Moment (GIM).*

“Halaman ini sengaja dikosongkan”

MOVING VEHICLE CLASSIFICATION BASED ON GEOMETRIC INVARIANT MOMENT

Name	***: Zamroji Hariyanto***
NRP	***: 1211 100 058***
Department	***: Mathematics***
Supervisor	***: Dr. Budi Setiyono, S.Si, MT***

Abstract

This study aims to classify and count the types of vehicles moving on a highway by using digital image processing. The object of this study is a video recording of the flow of vehicles. Then the election of ROI (Region of Interest) is used as the area classification and counting of moving vehicles. Furthermore, each ROI in the frame of the video is processed using background subtraction algorithm with Gaussian Mixture Model method to obtain the foreground and background images. Foreground image is filtered, eliminated its shadow, and performed morphological processes to eliminate noise in the image. Then the results are analyzed blob to get a part of the pixel which is the object moves. The next stage, each object obtained is classified by the Geometric invariant Moment. Each object will be counted the first of invariant moment (ϕ_1) *and its widht, then it will be used as a parameter for vehicle classification. If the value of the parameter is in the interval, it will be recognized as type that is defined on the interval. The results of software testing can classify vehicles in three types of vehicles are motorcycle, car, and truck/bus with the highest level of accuracy is 94.27%.*

Keywords: *Vehicle Counting, Vehicle Classification, digital image processing, a Gaussian Mixture Model (GMM), Geometric invariant Moment (GIM).*

"Halaman ini sengaja dikosongkan"

KATA PENGANTAR

Segala Puji bagi Allah SWT Tuhan semesta alam yang telah memberikan karunia, rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: **"Klasifikasi Jenis Kendaraan Bergerak Berbasis *Geometric Invariant Moment*"** yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Studi S-1 pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan berkat kerjasama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis mengucapkan terima kasih kepada:

1. Dr. Budi Setiyono, S.Si, MT selaku dosen pembimbing yang senantiasa membimbing dengan sabar dan memberikan kritik dan saran dalam penyusunan Tugas Akhir ini.
2. Prof. Dr. Erna Apriliani, M.Si selaku Ketua Jurusan Matematika.
3. Drs. Suhud Wahyudi, M.Si selaku Dosen Wali.
4. Prof. DR. Basuki Widodo, M.Sc, Dr. Imam Mukhlash, S.Si, MT, Drs. Suharmadi, Dipl.Sc, M.Phil, dan Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku dosen penguji Tugas Akhir ini.
5. Dr. Chairul Imron, MI.Komp. selaku Koordinator Tugas Akhir.
6. Seluruh jajaran dosen dan staf jurusan Matematika ITS.
7. Teman-teman mahasiswa jurusan Matematika ITS

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Januari 2015

Penulis

special thanks to

Selama proses pembuatan Tugas Akhir ini, banyak pihak yang telah memberikan bantuan dan dukungan untuk penulis. Penulis mengucapkan terima kasih dan apresisasi secara khusus kepada:

1. Ayah Muryanto dan Ibu Hanifah tersayang yang senantiasa dengan ikhlas memberikan semangat, perhatian, kasih saying, doa, motivasi dan nasihat yang sangat berarti bagi penulis.
2. Adik Khoirul Amiroh, Alfiyatut Toyyibah, dan Latifatul Nur Azizah yang menjadi salah satu motivasi bagi penulis untuk segera menyelesaikan Tugas Akhir ini.
3. Andika, Marmel, Chacha, Nuril, Ely, Risa, Budi, Muna Mbak Nadhia, Mas Romi, dan teman- teman pejuang 111 lainnya yang tidak bisa disebutkan satu per satu yang telah bersama-sama penulis berjuang dalam suka dan duka.
4. Sahabat Genggong SSM yaitu Lusi, Veda, Dina, Riyani, Yongky, Wanda, Isman, Faing, Hesti dan Sahabat Kepompong yaitu Ade N, Nilam, Eni, dan Toni yang selalu menjadi tempat *curhat* dan pendengar yang baik bagi penulis serta tidak bosan dalam memberi semangat dan saran untuk menyelesaikan Tugas Akhir ini.
5. Irvan, Ade R, Tyara, Liyana, Hakam, Ilham, Fendi, Habib, Rifdy, Jamil, Bundo, Henny, Mila dan teman-teman angkatan 2011 yang tidak bisa disebutkan satu per satu, terimakasih atas segala bentuk semangat dan dukungannya kepada penulis.

Tentu saja masih banyak pihak lain yang turut andil dalam penyelesaian tugas akhir ini yang tidak bisa penulis sebutkan satu persatu. Semoga Allah membalas dengan balasan yang lebih baik bagi semua pihak yang telah membantu penulis. *Amin ya rabbal 'alamin*.

DAFTAR ISI

"Halaman ini sengaja dikosongkan"

DAFTAR TABEL

"Halaman ini sengaja dikosongkan"

DAFTAR GAMBAR

DAFTAR PUSTAKA

[1] Anonim. (2013). "Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis tahun 1987-2012". http://bps.go.id/tab_sub/view.php?kat=2&tabel=1&daftar=1&id_subyek =17&notab=12 , *posted:* 2013. Diakses pada tanggal 07-09-2014

[2] Anonim. (2013). "Pusjatan PU-Jepang Gelar Seminar Jalan dan Jembatan". http://www.pu.go.id/m/main /view/8796, *posted:* Aug 29 2013 2:49PM. Diakses pada tanggal 20-08-2014

[3] Anonim. (2013). "Kisah Edi Warsito: Hitung Kendaraan Pemudik Secara Manual".http://news.liputan6.com/read/662163/kisah-edi-warsito-hitung-kendaraan-pemudik-secara-manual, *posted*:10/08/2013 14:43. Diakses pada tanggal 20-08-20134.

[4] Anonim. (2013). "Ada Alat Canggih untuk Hitung Kendaraan di Posko Nagreg". http://ramadan.okezone.com/read/2013/08/05/335/847739/ada-alat-canggih-untuk-hitung-kendaraan-di-posko-nagreg, *posted*:05/08/2013 15:40. Diakses pada tanggal 20-08-2014

[5] Lazuardi, R.A.F. (2014). "Penghitungan Kendaraan Bergerak Berbasis Algoritma Background Subtraction Menggunakan Metode Gaussian Mixture Model". **Tugas Akhir. Jurusan Matematika ITS**.

[6] Hu, M.K., (1962). "Visual Pattern Recognition by Moments Invariants". **IRE Trans. Information Theory Vol. 8**, Hal. 179- 87.

[7] Rizon, M., Yazid, H., Saad, P., Shakaff, A.Y.M., Saad, A.R., Mamat, M.R., Yaacob, S., Desa, H., dan Karthigayan,

M. (2006). “Object Detection using Geometric Invariant Moment”. **American Journal of Applied Sciences 2 (6)**, Hal. 1876-1878.

[8] Asaidi, H., Aarab, A., dan Bellouki, M. (2014). “Shadow Elimination and Vehicles Classification Approaches in Traffic Video Surveillance Context”. **Journal of Visual Languages and Computing Vol. 25**, Hal. 333-345.

[9] Khotanzad, A., dan Hong, Y.H. (1990). “Invariant Image Recognition by Zernike Moments”. **IEEE Trans. Pattern Analysis and Machine Intelligence Vol. 12**, Hal. 489-497.

[10] Gonzalez, R.C., dan Woods, R.E. (2002). “Digital Image Processing”. **United States of America: Tom Robbins Publisher**.

[11] Al Bovik. (2000). “Handbook of Image and Video Processing”. **San Diego: Academic Press Publisher**.

[12] Bharti, T. dan Tejinder. (2013). “Background Subtraction Techniques-Review”. **International Journal of Innovative Technology and Exploring Engineering, Vol. 2, Issue 3**, Hal. 166 - 168.

[13] C. Stauffer, W.E.L., dan Grimson. (1999). “Adaptive Background Mixture Models for Real-time Tracking”. **The Artificial Intelligence Laboratory, Cambridge**.

[14] Xu, L., Landabaso, J.L., dan Pardàs, M. (2005). “Shadow Removal with Blob-Based Morphological Reconstruction for Error Correction”. **IEEE Trans. Acoustics, Speech, and Signal Processing Vol. 2**, Hal. 729 – 732.

BIODATA PENULIS

Penulis memiliki nama lengkap Zamroji Hariyanto, lahir di Tulungagung pada tanggal 23 Maret 1993. Penulis berasal dari Kota Tulungagug, bertempat tinggal di Jalan Mastrip, Desa Serut Rt. 02 Rw. 02, Kec. Boyolangu, Kab. Tulungagung. Pendidikan formal yang pernah ditempuh yaitu SD Negeri 1 Serut Tulungagung, SMP Negeri 2 Tulungagung, SMA Negeri 1 Boyolangu Tulungagung. Kemudian, penulis melanjutkan studi di jurusan Matematika ITS, dengan bidang minat ilmu komputer. Selama menjadi mahasiswa penulis aktif di beberapa organisasi antara lain: HIMATIKA ITS dan UKM Koperasi Mahasiswa. Penulis juga mengikuti kepanitiaan acara besar yang ada di ITS diantaranya: ITS EXPO, OMITS, dan KNM-17. Selama menjadi mahasiswa penulis juga bekerja sebagai tentor mata pelajaran matematika, asisten dosen, dan *junior programmer*. Selama penulisan Tugas Akhir ini Penulis tidak lepas dari kekurangan, untuk itu penulis mengharapkan kritik, saran, dan pertanyaan mengenai Tugas Akhir ini yang dapat dikirimkan melalui *e-mail* ke zamroji.h@gmail.com.

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Jumlah kendaraan di Indonesia setiap tahun mengalami peningkatan, berdasarkan data dari Badan Pusat Statistik jumlah kendaraan bermotor pada tahun 2012 mencapai 94.373.324 unit[1]. Namun, pertumbuhan jumlah kendaraan tidak diimbangi dengan fasilitas jalan yang memadahi sehingga banyak timbul permasalahan-permasalahan lalu lintas.

Kemacetan merupakan salah satu permasalahan yang menjadi perhatian khusus bagi pemerintah saat ini. Dikarenakan kemacetan memiliki pengaruh besar dalam proses kinerja suatu usaha, distribusi barang, efisiensi waktu, keberhasilan usaha, tingkat keselamatan, dan aktivitas kerja yang lain. Untuk menghindari kemacetan, salah satunya dengan membangun infrastruktur jalan. Agar pembangunan infrastruktur jalan dapat efektif dan sesuai dengan beban yang melintasinya, maka diperlukan informasi mengenai volume kendaran sesuai dengan jenisnya pada suatu jalan.

Saat ini, metode yang digunakan untuk mengukur volume lalu lintas di Indonesia kurang akurat[2], dikarenakan perhitungan jumlah kendaraan masih menggunakan tenaga manusia yang dibantu dengan alat hitung manual[3]. Hingga Agustus 2013, hanya ada tiga kota yang baru menerapkan teknologi otomatis untuk melakukan pekerjaan tersebut. Alat tersebut dinamakan *Automatic Traffic Counter* yang baru diuji coba oleh Dinas Perhubungan Kabupaten Bandung di Nagreg, Dinas Perhubungan Provinsi di Sadang, dan Kementrian Perhubungan di Indramayu untuk menghitung arus mudik dan arus balik Lebaran tahun 2013[4].

Perhitungan kendaraan dengan cara manual rentan akan terjadinya kesalahan karena faktor kondisi manusia, suasana, dan area kerja. Sehingga diperlukan metode lain yang dapat melakukan perhitungan secara otomatis sehingga menghasilkan

informasi yang lebih akurat. Metode pengolahan citra merupakan salah satu metode yang dapat digunakan untuk mengolah informasi dari gambar atau video. Terdapat beberapa penelitian yang telah dilakukan untuk menghitung kendaraan bergerak berbasis citra digital. Salah satu diantaranya adalah penelitian yang dilakukan oleh R. Arif Firdaus Lazuardi [5].

Dalam penelitian R. Arif, perangkat lunak yang dibangun mampu menghitung kendaraan dengan tingkat keberhasilan rata-rata pada waktu pagi hari adalah 94,51% dan siang hari 95,28%[5]. Namun, penelitian tersebut belum dapat mengklasifikasikan kendaraan berdasarkan jenisnya.

Penelitian selanjutnya adalah *Geometric Invariant Moment (GIM)* yang pertama kali dikenalkan oleh Hu[2]. GIM digunakan untuk mengekstrak ciri yang bersifat *Rotation Scale Translation* (RST)-*invariant* dari sebuah citra, dengan kata lain ciri yang dihasilkan tidak berubah terhadap perlakuan rotasi, penskalaan, dan translasi. GIM telah sukses diterapkan dalam pengenalan objek, dalam penelitian yang berjudul *Object Detection using Geometric Invariant Moment* dengan akurasi 90%[7], dan pada penelitian yang dilakukan oleh Asaidi dkk dengan akurasi 96,96%[8] serta dalam pengenalan pesawat terbang, klasifikasi pola, dan citra radar untuk pencocokan citra optik[9]. Asaidi dkk pada penelitian [8] menyatakan bahwa metode ini memberikan hasil yang lebih baik dibandingkan metode analisis tekstur dan metode tradisonal yang hanya memperhatikan ukuran dan sebuah *single frame*. Dari penjabaran tersebut, penulis memilih untuk melakukan penelitian mengenai pengklasifikasian jenis kendaraan menggunakan berbasis *Geometric Invariant Moment (GIM)*.

1.2 Rumusan Masalah

Rumusan masalah dari Tugas Akhir ini adalah:

1. Bagaimana menganalisa nilai *Geometric Invariant Moment* dari citra objek kendaraan untuk dijadikan sebagai parameter klasifikasi jenis kendaraan bergerak?
2. Bagaimana mensintesa metode *Gaussian Mixture Model*, Analisis *Blob,* dan *Geometric Invariant Moment* untuk mengklasifikasikan jenis kendaraan bergerak?
3. Bagaimana merumuskan algoritma untuk mengklasifikasikan jenis kendaraan bergerak berbasis *Geometric Invariant Moment*?
4. Bagaimana mengembangkan perangkat lunak yang mampu mengolah informasi dari video digital untuk mengklasifikasikan jenis kendaraan bergerak berbasis *Geometric Invariant Moment*?

1.3 Batasan Masalah

Untuk menfokuskan penelitian Tugas Akhir ini, maka perlu diambil beberapa batasan masalah sebagai berikut:

1. Data yang digunakan adalah rekaman video arus lalu lintas kendaraan di Jalan Embong Malang, Surabaya pada saat pagi dan siang.
2. Sudut pandang yang diambil ketika merekam adalah dari atas dengan jangkauan pandangan seluruh jalan arus satu arah.
3. Menggunakan kamera digital dengan resolusi 320x240 dan *framerate* minimal 25 *frame per second* (fps).
4. Informasi digital yang diproses dilakukan secara *off line*.
5. Mengklasifikasikan jenis kendaraan dalam jenis sepeda motor, mobil, dan truk/bus.

1.4 Tujuan Penelitian

Tujuan dari Tugas Akhir ini adalah :

1. Menganalisa nilai *Geometric Invariant Moment* dari citra objek kendaraan untuk dijadikan sebagai parameter klasifikasi jenis kendaraan bergerak.

2. Mensintesa metode *Gaussian Mixture Model*, Analisis *Blob,* dan *Geometric Invariant Moment* untuk mengklasifikasikan jenis kendaraan bergerak
3. Merumuskan algoritma untuk mengklasifikasikan jenis kendaraan bergerak berbasis *Geometric Invariant Moment.*
4. Mengembangkan perangkat lunak yang dapat mengolah informasi dari video digital untuk mengklasifikasikan jenis kendaraan bergerak berbasis *Geometric Invariant Moment.*

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari Tugas Akhir ini adalah:

1. Sistem yang telah dibangun dapat memberikan informasi tentang jumlah dan jenis kendaraan yang melaju di jalan satu arah sehingga dapat dijadikan bahan pertimbangan bagi instansi pemerintah terkait untuk merumuskan kebijakan rekayasa lalu lintas.
2. Memberikan kontribusi bagi dunia penelitian khususnya dalam pengembangan perangkat lunak untuk klasifikasi jenis kendaraan.

1.6 Sistematika Penulisan

Sistematika Penulisan Tugas Akhir ini dibagi menjadi lima bab. Isi dari masing- masing bab tersebut adalah sebagai berikut :

1. Bab I Pendahuluan
   Pada bab ini dibahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat dari penelitan yang dilakukan, serta sistematika penulisan Tugas Akhir.
2. Bab II Studi Pustaka
   Pada bab ini diuraikan beberapa teori pendukung yang berasal dari jurnal dan buku yang berkaitan atau yang melandasi pembahasan pada penelitian ini untuk membantu menyelesaikan permasalahan Tugas Akhir.
3. Bab III Metodologi
   Pada bab ini berisi tentang langkah-langkah yang digunakan untuk menyelesaikan Tugas Akhir.

4. Bab IV Perancangan dan Implementasi Sistem
   Pada bab ini akan dibahas mengenai analisis dan desain rancangan sistem dimulai dari pembahasan proses pengambilan data masukan, pengolahan data masukan dengan algoritma *Background Subtraction* dan metode *Gaussian Mixture Model* serta metode *Geometric Invarian Moment* untuk pengklasifikasian jenis, serta penjelasan mengenai cara untuk mendapatkan data keluaran yang sesuai dengan tujuan dari penelitian Tugas Akhir ini. Kemudian implementasi sistem dalam bentuk perangkat lunak untuk memproses informasi dari rekaman video digital arus kendaraan
5. Bab V Pengujian dan Pembahasan
   Bab ini menyajikan pembahasan hasil uji coba program dari beberapa rekaman video arus lalu lintas. Kemudian dicatat sebagai bahan untuk merumuskan beberapa kesimpulan dan saran dari Tugas Akhir ini.
6. Bab VI Kesimpulan dan Saran
   Bab ini berisi kesimpulan dari hasil penelitian yang telah dilakukan dan saran untuk pengembangan dari penelitian ini.

“Halaman ini sengaja dikosongkan”

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian dengan tema yang terkait dengan Tugas Akhir ini yaitu penelitian dari Asaidi, dkk[8] yang berjudul "*Shadow Elimination and Vehicles Classification Approaches in Traffic Video Surveillance Context*". Dalam penelitian Asaidi, dkk ini menggunakan *Geometric Invariant Moment* dari objek citra kendaraan untuk mengklasifikasikan jenis dari kendaraan tersebut. Penelitian ini berhasil mengklasifikasikan jenis kendaraan dengan tingkat akurasi 96,96%. Penelitian berikutnya, mengenai pengenalan objek menggunakan *Geometric Invariant Moment* oleh Mohamed Rizon, dkk [7] telah berhasil mengenali objek gambar kelapa dan palem dengan akurasi 90% dengan menggunakan deteksi tepi Sobel pada segmentasi citra.

R. Arif Firdaus Lazuardi[5] telah melakukan penelitian untuk menghitung kendaraan menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model*. Dalam penelitiannya, perangkat lunak yang dibangun mampu menghitung kendaraan dengan tingkat keberhasilan rata-rata pada waktu pagi hari adalah 94,51% dan siang hari 95,28%. Namun, perangkat lunak yang dibangun belum dapat mengklasifikasikan jenis kendaraan.

Tugas akhir ini akan mengembangkan penelitian dari R. Arif, dengan menambahkan fitur klasifikasi jenis kendaraan dalam perangkat lunak yang dibangun. Klasifikasi jenis kendaraan pada penelitian Tugas Akhir ini berbasis *Geometric Invariant Moment* yang telah berhasil digunakan untuk pengenalan objek pada penelitian sebelumnya. Pada subbab selanjutnya akan dijelaskan dasar teori yang digunakan dalam penelitian ini.

2.2 Pengertian Citra

Citra menurut Kamus Besar Bahasa Indonesia memiliki makna rupa, gambar, atau gambaran. Sedangkan menurut kamus Webster citra adalah suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda. Citra terbagi menjadi dua yaitu citra diam dan citra bergerak. Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan, citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan pada mata kita sebagai gambar yang bergerak.

Pada awalnya citra yang dikenal manusia berbentuk citra kontinu. Suatu representasi objek yang dihasilkan dari sistem optik yang menerima sinyal analog dan dinyatakan dalam bidang dua dimensi. Nilai cahaya yang ditransmisikan pada citra kontinu memiliki rentang nilai yang tak terbatas. Contoh dari citra kontinu adalah mata manusia dan kamera analog.

Sebuah citra analog tidak dapat direpresentasikan secara langsung oleh komputer. Oleh sebab itu dilakukan sebuah proses untuk merubah nilai-nilai yang ada pada citra analog agar komputer dapat membaca dan menerjemahkan informasi yang terdapat pada citra analog. Hasil dari pemrosesan tersebut dinamakan sebagai citra digital[5].

2.3 Citra Digital

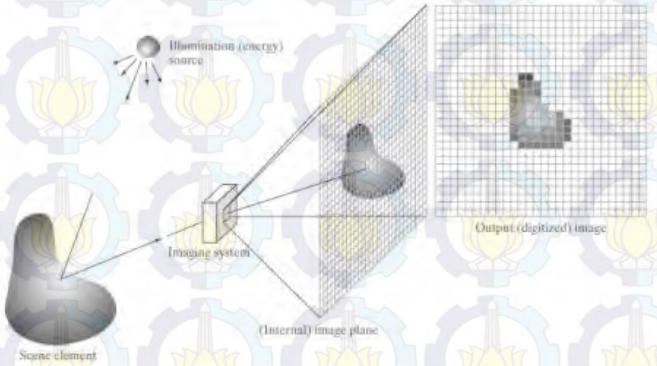
Dalam buku *Digital Image Processing* yang ditulis oleh Rafael C. Gonzalez dan Richard E. Woods[10], menjelaskan bahwa citra digital merupakan fungsi dua dimensi yang dapat dinyatakan dengan fungsi $f(x,y)$, dimana x dan y merupakan titik koordinat spasial. Sedangkan amplitudo dari fungsi f pada sembarang koordinat (x,y) merupakan nilai intensitas cahaya, yang merupakan representasi dari warna cahaya yang ada pada citra analog. Citra digital adalah suatu citra dimana (x,y) dan nilai intensitas dari f terbatas (*discrete quantities*), dan telah dilakukan proses digitalisasi spasial dan digitalisisi kuantitas.

2.3.1 Digitalisasi Spasial (*Sampling*)

Sampling merupakan proses pengambilan informasi dari citra analog yang memiliki panjang dan lebar tertentu untuk membaginya ke beberapa blok kecil. Blok-blok tersebut disebut sebagai piksel. Sehingga citra digital yang lazim dinyatakan dalam bentuk matriks memiliki ukuran *M* x *N* dengan *M* sebagai baris dan *N* kolom. Bisa juga disebut sebagai citra digital yang memiliki *M* x *N* buah piksel. Notasi matriks citra digital dapat dinyatakan sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,\ N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \quad (2.1)$$

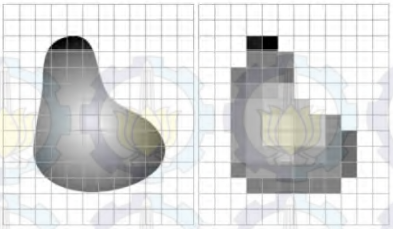
Proses *sampling* citra analog ke citra digital ditampilkan pada Gambar 2.1 di bawah ini:



Gambar 2.1. Proses *sampling* citra dari citra analog ke citra digital. (sumber: *"Invariant Image Recognition by Zernike Moments"*[10])

2.3.2 Digitalisasi Intensitas (Kuantisasi)

Kuantisasi adalah proses pemberian nilai derajat keabuan di setiap titik piksel yang merupakan representasi dari warna asli dari citra analog. Rentang nilai keabuan adalah 0 – 255.



Gambar 2.2. Proses sampling dan kuantisasi. (a) Gambar citra analog yang ditempatkan pada sensor *array*. (b) Gambar citra setelah proses *sampling* dan kuantisasi, tiap piksel pada citra *b* memiliki derajat keabuan masing-masing.
(sumber: *"Invariant Image Recognition by Zernike Moments"*[10])

2.4 Video Digital

Video adalah teknologi untuk menangkap, merekam, memproses, menyimpan, dan merekonstruksi suatu urutan dari beberapa citra. Alan C. Bovik menjelaskan bahwa video digital merupakan hasil *sampling* dan kuantisasi dari video analog[11]. Secara mendasar, tidak ada perbedaan proses sampling dan kuantisasi antara citra digital dan video digital.

Bagaimanapun juga, video analog yang kita lihat sehari-sehari seperti tampilan pada TV analog, sebenarnya bukan sesuatu yang benar-benar kontinu, melainkan terdiri dari beberapa frame yang ditampilkan dengan kecepatan tertentu. Setiap *frame* merupakan citra analog dan kecepatan untuk menampilkan citra-citra yang ada disebut sebagai *frame rate* dengan satuan fps (*frame per second*). Jika *frame rate* cukup tinggi, maka akan telihat sebagai rangkaian yang kontinu sehingga tercipta ilusi gerak yang halus.

Video analog dapat dinyatakan dengan fungsi I(x,y,t), dimana (x,y) adalah nilai kontinu dari fungsi I dan t menyatakan waktu. Sebenarnya tampilan video analog di TV maupun monitor merupakan representasi dari fungsi sinyal elektrik satu dimensi V(t) yang terdiri dari beberapa citra analog I(x,y,t) dengan jumlah citra (x,y) tertentu dan waktu (t) tertentu. Proses pemisahan video ke beberapa unit frame citra disebut sebagai *scanning*.

2.5 Algoritma *Background Subtraction*

Background Subtraction (BS) digunakan untuk mendapatkan objek yang bergerak pada serangkaian *image*. Objek yang bergerak dapat diidentifikasi dengan melakukan pengurangan antara *frame* pada waktu t dengan *background* model (latar belakang model). Kemudian nilai citra hasil pengurangan tersebut dibandingkan dengan nilai ambang batas (*threshold*) sesuai dengan metode yang dipakai untuk membangun *foreground image*[12].

Secara matematis proses tersebut dapat ditulis sebagai berikut

$$|I(x,y,t) - B(x,y,t)| = d(x,y,t) \quad (2.2)$$

$$f(x,y,t) = \begin{cases} 1 & jika\ d(x,y,t) \geq \tau \\ 0 & selainnya \end{cases} \quad (2.3)$$

dengan $I(x,y,t)$ adalah citra pada waktu t, $B(x,y,t)$ adalah model *background* pada waktu t, $d(x,y,t)$ adalah citra hasil pengurangan I dan B pada waktu t, $f(x,y,t)$ adalah *foreground image* (gerak yang dideteksi) dan τ adalah *threshold*. Dalam penelitian ini akan digunakan nilai τ =254.

Gambar 2.3. Proses *background subtraction* dengan metode *frame differencing* untuk mendapatkan *foreground image*.
(Sumber: *"Background Subtraction Techniques-Review"* [12])

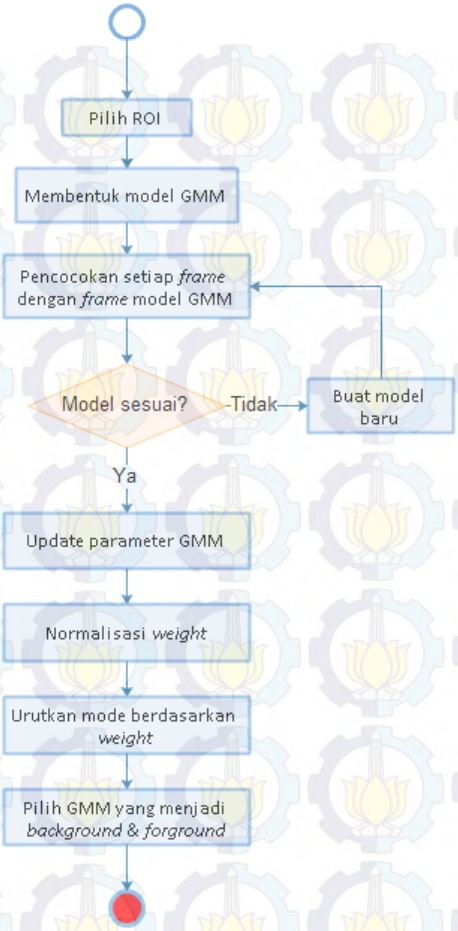
Ada berbagai macam metode untuk menginisialisasi *background* model. Salah satunya adalah metode *Gaussian Mixture Model*.

2.6 Metode *Gaussian Mixture Model*

Gaussian Mixture Model (GMM) merupakan metode yang tepat untuk berbagai kondisi yang terdapat pada citra, seperti *background* citra yang selalu statis, multimodal, maupun yang mengandung *noise* (gangguan atau objek yang tidak diinginkan terdapat pada citra).

Pada proses GMM dibutuhkan algoritma *clustering* untuk mengelompokkan piksel-piksel mana saja yang termasuk *foreground* atau *background*. Pada umumnya metode *clustering* yang digunakan adalah *Expectation Maximization* dan K-means. Chris Stauffer dalam penelitiannya menggunakan algoritma *clustering* K-means dalam mencari dan meng-update model *background* karena metode tersebut memiliki kecepatan proses yang lebih cepat dan hasil yang cukup baik[13].

GMM merupakan tipe *density model* yang terdiri dari komponen fungsi-fungsi Gaussian. Komponen fungsi tersebut terdiri dari *weight* yang berbeda untuk menghasilkan *multi model density*. Model-model GMM terbentuk dari data warna piksel berdasarkan waktu. Hasil model tersebut akan menjadi 2 bagian, model yang mencerminkan *background* dan model non-*background*. Jumlah model GMM yang digunakan mempengaruhi jumlah model *background*. Semakin besar jumlah model GMM yang dipakai semakin banyak model background yang dimiliki suatu piksel. R. Arif telah melakukan Uji coba terhadap parameter-parameter Gaussian menggunakan video rekaman yang berdurasi 30 detik. Sehingga diperoleh kesimpulan pada penelitiannya yaitu dengan menggunakan jumlah model antara 3 s.d 7 dan *learning rate* 0.001, 0.005, atau 0.01 dapat diperoleh model menyerupai *background* sebenarnya[5]. Berikut disajikan diagram alir proses GMM oleh R. Arif:



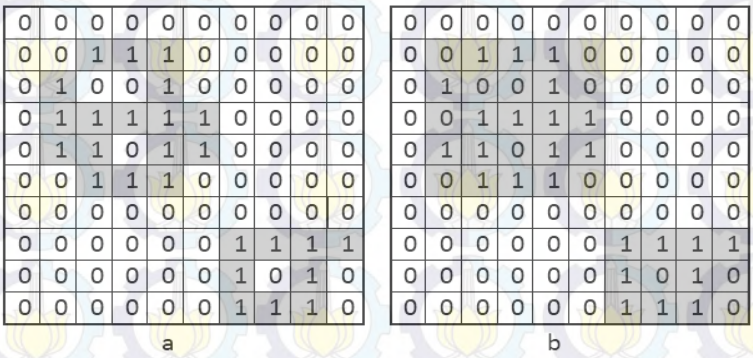
Gambar 2.4. Diagram alir proses GMM.

2.7 Metode Analisis *Blob*

Analisis *blob* ini menggunakan metode *connected component*, dimana di setiap kumpulan piksel yang tingkat keabuannya bernilai satu, dikategorikan sebagai satu objek. Setiap objek yang terdeteksi akan diberi label untuk mempermudah pengolahan. Pada Gambar 2.5 menjelaskan tentang pemilihan area *blob* yang terdeteksi. Dari analisis *blob* tersebut akan diperoleh informasi tentang *centroid*, luas area, tinggi, dan lebar sebuah objek dari bentuk *rectangle*. *Centroid* dari objek *blob* $b(x,y)$ dengan ukuran MxN dapat diperoleh menggunakan persamaan (2.4).

$$\bar{x} = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} x\, b(x,y)}{\sum_{x=1}^{M}\sum_{y=1}^{N} b(x,y)}, \bar{y} = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} y\, b(x,y)}{\sum_{x=1}^{M}\sum_{y=1}^{N} b(x,y)} \qquad (2.4)$$

Dengan $(\bar{x}, \bar{y})$ adalah centroid dari objek *blob*[7].



Gambar 2.5. Bagian a adalah representasi citra ukuran 10x10 piksel dan citra dengan nilai piksel 1 merupakan representasi dari objek (*foreground*). Sedangkan bagian b merupakan *blob* yang terdeteksi.

(sumber: "Penghitungan Kendaraan Bergerak Berbasis Algoritma Background Subtraction Menggunakan Metode Gaussian Mixture Model" [5])

2.8 *Shadow Removal*

Bayangan (*shadow*) merupakan bidang yang terbentuk akibat hilangnya cahaya karena mengenai suatu objek yang tidak dapat ditembus oleh cahaya tersebut. Terdapat dua jenis bayangan yaitu *self-shadow* dan *cast-shadow*[14].

Gambar 2.6. Jenis-jenis bayangan (*shadow*)

Bayangan berpengaruh dalam perhitungan dan pengenalan jenis kendaraan, oleh karena itu diperlukan proses untuk menghilangkan banyangan (*shadow removal*) sebelum dilakukan perhitungan dan pengklasifikasian.

Texture-based detection[14] merupakan salah satu metode yang digunakan untuk menghilangkan bayangan. Tekstur *shadow* memiliki kemiripan dengan tekstur *background*, sehingga untuk menentukan daerah yang terkena bayangan dapat dilakukan dengan cara menghitung jarak e*cluidean* antara citra *background* dengan citra *foreground*. Jika jarak tersebut lebih kecil dari batas tertentu, maka piksel tersebut merupakan bagian dari daerah bayangan.

2.9 *Geometric Invariant Moment*

Geometric Invariant Moment (GIM) pertama kali dikenalkan oleh Hu[6]. GIM dipilih karena dapat mengekstrak ciri yang bersifat *Rotation Scale Translation* (RST)-*invariant* dari sebuah citra atau dengan kata lain ciri yang dihasilkan tidak berubah terhadap perlakuan rotasi, penskalaan, dan translasi. Momen dari suatu fungsi $g(x, y)$ kontinu didefinisikan sebagai berikut:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q g(x,y)dxdy \qquad (2.5)$$

dengan $p, q = 0,1,2, \ldots$

x dan y merupakan koordinat spasial pada fungsi $g(x, y)$ sedangkan p dan q merupakan nilai order dari momen.

Untuk momen dua dimensi dari suatu citra digital dengan ukuran MxN didefinisikan sebagai berikut:

$$m_{pq} = \sum_{x=1}^{M}\sum_{y=1}^{N} x^p y^q g(x,y) \qquad (2.6)$$

Selanjutnya momen pusat (*central moments*) didefinisikan sebagai:

$$\mu_{pq} = \sum_{x=1}^{M}\sum_{y=1}^{N} (x-\bar{x})^p (y-\bar{y})^q g(x,y) \qquad (2.7)$$

dengan $\bar{x} = \frac{m_{10}}{m_{00}}$ dan $\bar{y} = \frac{m_{01}}{m_{00}}$.

Kemudian apabila momen pusat dinormalisasi akan menjadi *normalized central moments* sebagai berikut:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}{}^{\gamma}}, \qquad \gamma = \frac{p+q}{2} + 1 \qquad (2.8)$$

Secara khusus, Hu mendefinisikan tujuh nilai momen invarian, yang diturunkan dari orde kedua dan ketiga *normalized central moments*. Tujuh momen dinyatakan sebagai berikut:

$\phi_1 = \eta_{20} + \eta_{02}$

$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$

$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$

$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (2.9)$$

Ketujuh momen ini nantinya akan dianalisa terlebih dahulu untuk menentukan momen yang dapat dijadikan sebagai parameter untuk pengenalan jenis kendaraan.

"Halaman ini sengaja dikosongkan"

BAB III
METODOLOGI

Terdapat beberapa tahapan dalam penyusunan Tugas Akhir ini, antara lain:

3.1 Studi Literatur dan Perekaman Video

Pada tahap pertama ini akan dilakukan pengajian tentang pengolahan video digital, pengolahan citra, metode *Gaussian Mixture Model,* metode penghilangan bayangan (*shadow removal*), dan metode *Geometric Invariant Moment.* Studi ini dilakukan dengan membaca buku, jurnal, ataupun artikel yang terkait serta melakukan diskusi dengan dosen pembimbing dan dosen yang memiliki keahlian dalam bidang ini.

3.2 Analisis dan Desain Sistem

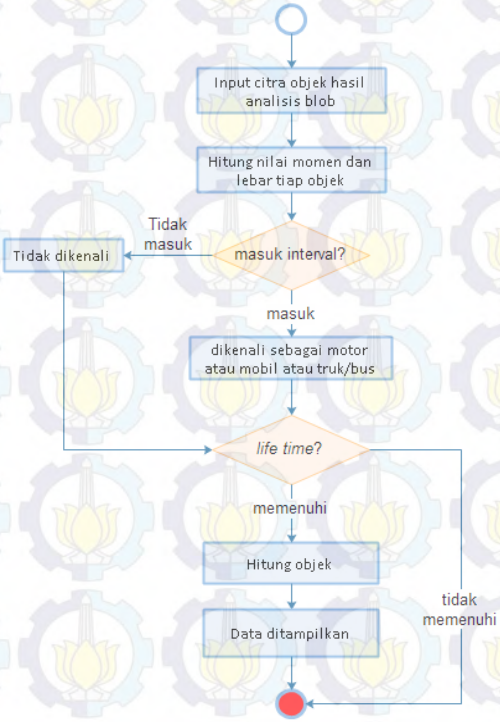
Pada tahap kedua ini akan dilakukan analisis video dan penentuan parameter-parameter apa yang akan dibutuhkan dalam pembuatan program. Kemudian dibuat desain sistem dari program sesuai dengan hasil analisis.

Berikut beberapa tahapan untuk mengolah rekaman video *off-line* arus kendaraan:

a. Sistem ini memiliki inputan berupa video digital. Pemilihan area ROI (*Region of Interest*) pada rekaman video yang digunakan sebagai area penghitungan dan klasifikasi.
b. *Scanning* untuk mendapatkan setiap *frame* dari video.
c. *Background Subtraction* menggunakan metode GMM untuk segmentasi setiap *frame* menjadi citra *background* dan *foreground.*
d. Penghilangan bayangan dan *filtering* agar citra bebas dari *noise* sehingga dapat diolah dengan mudah.
e. Analisis *blob* pada citra *foreground* untuk mendeteksi objek pada citra. Selanjutnya proses *tracking* untuk menentukan apakah objek pada *frame* ke-*t* dengan *frame* ke- *t+1* merupaakan objek yang sama.

f. Objek-objek itu akan diidentifikasi jenisnya menggunakan GIM, yang selanjutnya dihitung berdasarkan jenisnya.

Blok diagram alir klasifikasi menggunakan GIM dapat dilihat pada Gambar 3.1.



Gambar 3.1. Diagram alir klasifikasi kendaraan dengan metode GIM.

3.3 Implementasi Sistem

Pada tahap ketiga akan dilanjutkan dengan implementasi sistem dalam bentuk perangkat lunak sesuai dengan hasil analisis dan desain sistem. Akan dibuat desain *interface* yang menarik dan *user friendly* untuk memudahkan dan membuat nyaman pengguna.

3.4 Uji Coba dan Pembahasan

Pada tahap keempat ini dilakukan pengujian terhadap perangkat lunak yang telah dibangun. Uji coba ini meliputi uji coba parameter GIM untuk menentukan momen yang dapat digunakan sebagai parameter klasifikasi kendaraan. Kemudian dilanjutkan dengan uji coba klasifikasi kendaraan, data hasil klasifikasi dari program akan dibandingkan dengan klasifikasi data sebenarnya. Kemudian dihitung tingkat akurasi dari program menggunakan persamaan (3.1). Selain itu akan dilakukan evaluasi sehingga dapat membenahi hal-hal yang masih kurang dan mengatasi error yang ditemukan.

$$PK = \frac{JP}{JS} \times 100\% \qquad (3.1)$$

dengan PK adalah prosentase keberhasilan, JT adalah jumlah kendaraan yang terdeteksi dengan benar, dan JS adalah jumlah kendaraan sebenarnya.

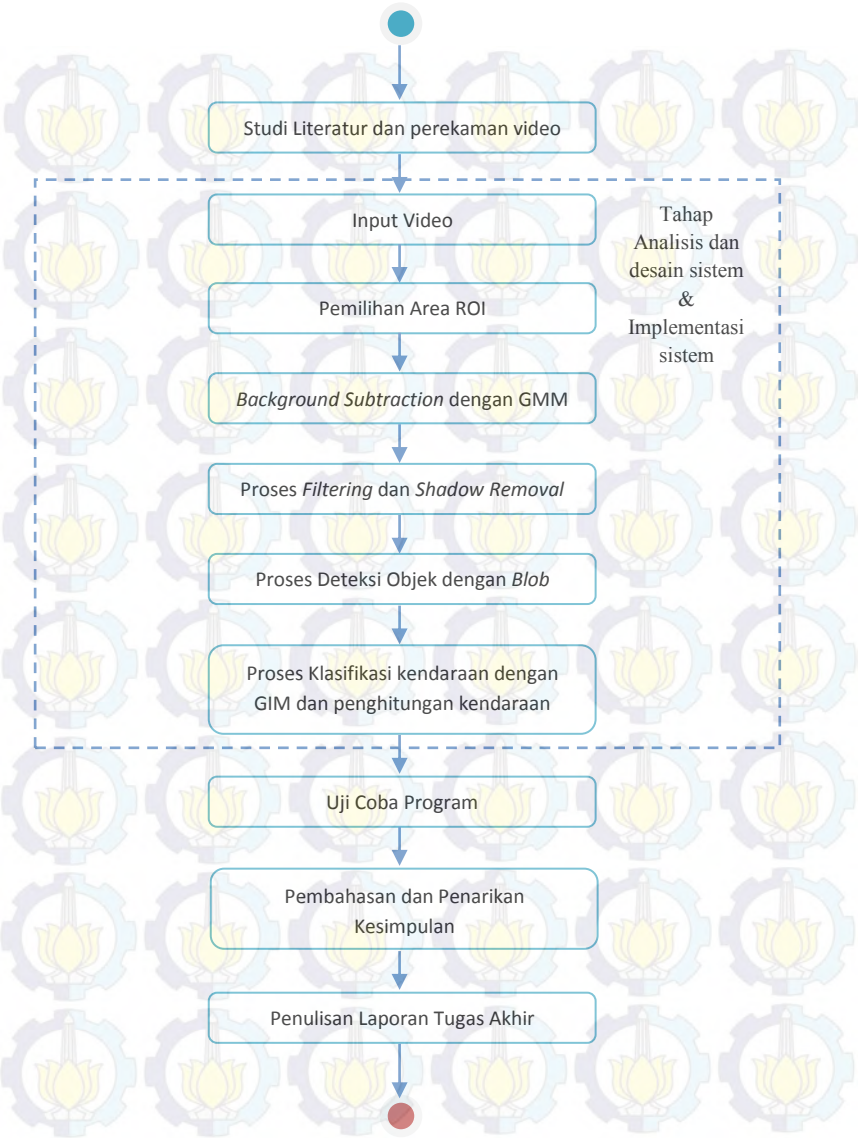
3.5 Penarikan Kesimpulan

Tahap kelima ini akan ditarik kesimpulan dari hasil penelitian, uji coba, dan pembahasan serta disampaikan saran untuk pengembangan berikutnya.

3.6 Penulisan Laporan Tugas Akhir

Tahap keenam ini merupakan tahap yang terakhir. Penulis akan menuliskan semua hasil yang telah didapatkan selama pengerjaan tugas akhir ini.

Adapun diagram dari tahap penelitian ini dapat dilihat pada Gambar 3.2 berikut.



Gambar 3.2. Diagram Metodologi Penelitian**.**

BAB IV
PERANCANGAN DAN IMPLEMENTASI SISTEM

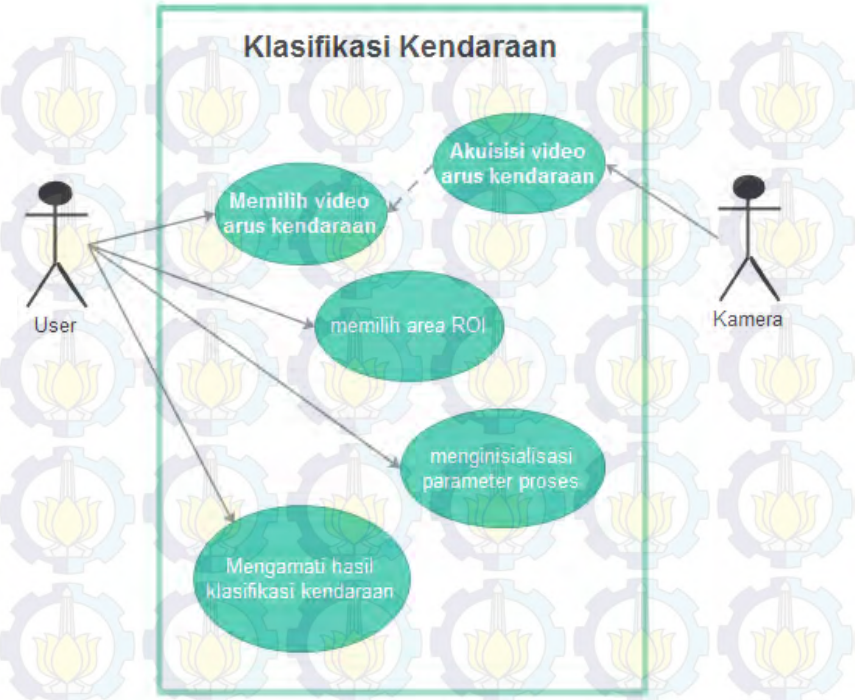
Pada bab ini akan dibahas mengenai perancangan dan implementasi sistem dimulai dari pembahasan proses pengambilan data masukan, pengolahan data masukan dengan algoritma *Background Subtraction* dan metode *Gaussian Mixture Model* serta metode *Geometric Invarian Moment* untuk pengklasifikasian jenis, serta penjelasan mengenai cara untuk mendapatkan data keluaran yang sesuai dengan tujuan dari penelitian Tugas Akhir ini.

4.1 Analisis Sistem

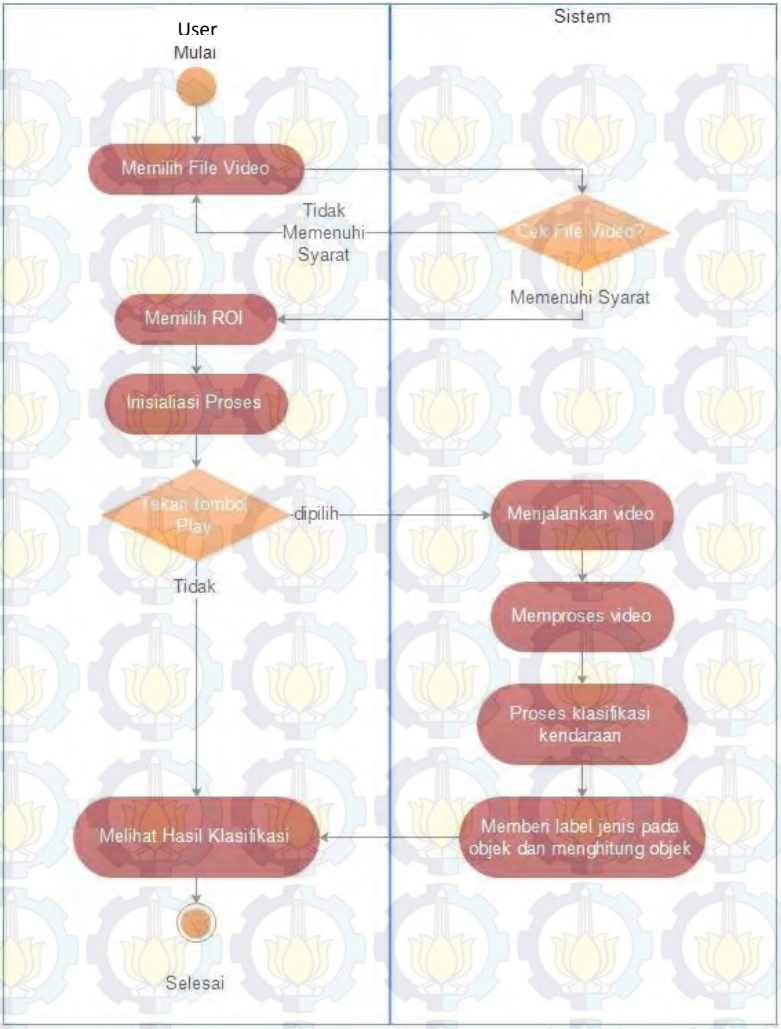
Untuk membangun perangkat lunak yang dapat menghitung dan mengklasifikasikan jenis kendaraan bergerak berbasis video digital. Maka diperlukan sistem yang dapat mengolah data masukan berupa rekaman arus kendaraan secara *real time*, pada kasus Tugas Akhir ini menggunakan data masukan berupa video digital arus kendaraan secara *offline*. Dengan mengkombinasikan algoritma *Background Subtraction* dengan metode *Gaussisan Mixture Model*[5] untuk mendapatkan objek kendaraan dalam video dan metode *Geometric Invarian Moment* untuk mengklasifikasikan jenis objek kendaraan maka akan tercipta sistem yang sesuai dengan tujuan dari Penelitian Tugas Akhir ini.

4.1.1 Analisis Sistem Perangkat Lunak

Perangkat lunak yang akan dibangun dapat digunakan oleh dinas perhubungan untuk mengamati kondisi arus lalu lintas di suatu jalan sehingga dapat menganalisa kebutuhan infrastruktur jalan tersebut. Selain itu masyarakat juga dapat memperoleh informasi kondisi lalu lintas pada suatu jalan. *Use Case Diagram* dari perangkat lunak ini disajikan pada Gambar 4.1 dan *Activity Diagram* pada Gambar 4.2.



Gambar 4.1. *Use Case Diagram* Perangkat Lunak Klasifikasi Kendaraan.

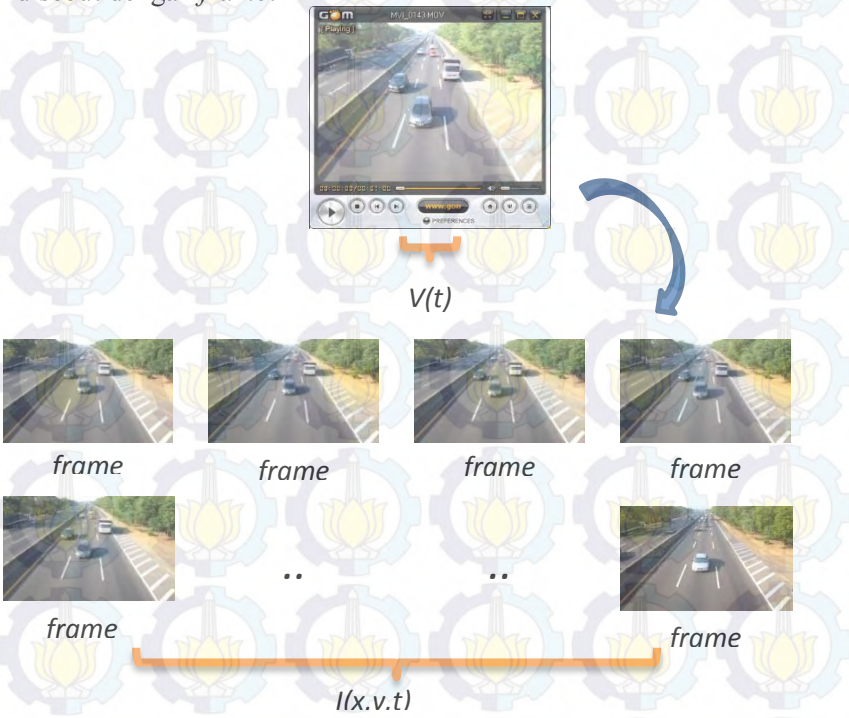


Gambar 4.2. *Activity Diagram* Perangkat Lunak Klasifikasi Kendaraan.

Sistem perangkat lunak yang dibangun ini memiliki beberapa tahapan sebagai berikut :

a. Akuisisi video

Data masukan berupa rekaman video digital *offline* arus kendaraan yang kemudian di-*scanning*. *Scanning* adalah proses pemecahan video menjadi beberapa rangkaian citra yang sering disebut dengan *frame*.



Gambar 4.3. Proses *Scanning* Video.

Pengambilan video arus lalu lintas kendaraan di Jalan Embong Malang Surabaya. Video diambil menggunakan kamera digital Canon 12.1 Mega Piksel. Video yang diambil berdasarkan ketentuan berikut:

- Sudut pandang yang diambil ketika merekam adalah dari atas dengan jangkauan pandangan seluruh jalan arus satu arah.

- Rekaman video memiliki resolusi 320x240 pixel dan *framerate* 25 *frame per second* (fps).

b. Penentuan ROI (*Region of Intereset*)

ROI (*Region of Intereset*) akan digunakan sebagai area penghitungan dan klasifikasi kendaraan bergerak. Jika objek kendaraan melewati area ROI dengan kecepatan lebih dari 5 fps, maka kendaraan tersebut ditangkap sebagai objek selain itu dianggap *noise* (bukan objek).

Gambar 4.4. Citra dengan ROI (daerah didalam kotak biru).

c. Segmentasi citra

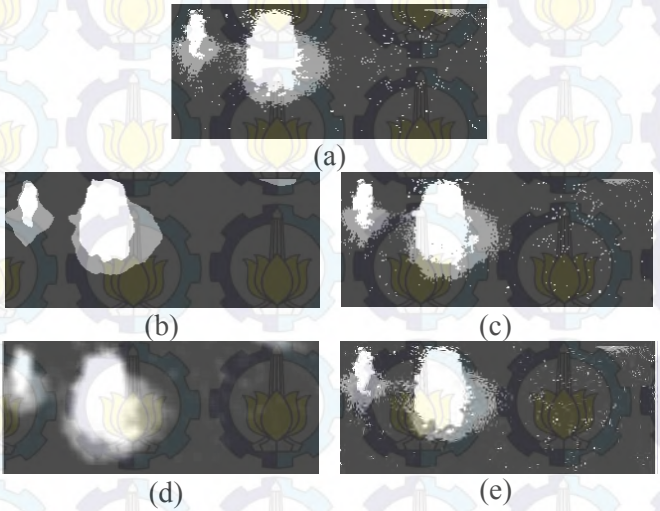
Kemudian sistem membagi *frame* menjadi citra *background* dan citra *foreground* dengan menggunakan algoritma *Background Subtraction* dengan metode *Gaussisan Mixture Model*[5]. Citra *background* adalah citra yang menjadi latar belakang dari suatu objek yang bergerak. Sedangkan *foreground* adalah gambaran objek bergerak yang terdeteksi.

(a) (b) (c)

Gambar 4.5. Citra hasil *Background Subtraction* dengan metode *Gaussian Mixture Model*. (a) citra asli, (b) citra *foreground*, dan (c) citra *background*.

d. Proses *filtering*

Citra *foreground* yang diperoleh di-*filter* untuk menghilangkan beberapa *noise* yang ada pada citra. Sehingga citra akan lebih mudah untuk diolah dan objek kendaraan akan tampak lebih jelas. *Noise* ini merupakan citra yang tertangkap akibat pengaruh cahaya, hembusan angin, bayangan yang muncul dari objek, serta gangguan-gangguan lain yang tertangkap oleh kamera. Terdapat beberapa metode *filtering* yang akan digunakan pada sistem, antara lain; metode median, metode Gaussian, metode blur, dan metode bilateral. Hasil *filtering* dapat dilihat pada Gambar 4.6.



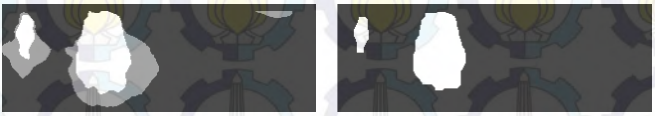
Gambar 4.6. Citra Hasil *Filtering*. (a) citra asli, (b) dengan metode median, (c) dengan metode gaussian, (d) dengan metode blur, dan (e) dengan metode bilateral.

Selanjutnya citra hasil *filtering* akan dilakukan penghilangan bayangan dan proses morfologi sehingga menghasilkan citra yang ditunjukkan pada Gambar 4.7. Penghilangan bayangan menggunakan *Texture-based detection*[14] dengan persamaan (4.1).

$$h(x,y,t) = \begin{cases} 1 & jika\ F(x,y,t) \geq \tau \\ 0 & selainnya \end{cases} \quad (4.1)$$

dengan $F(x,y,t)$ adalah citra *foreground* pada waktu t, $h(x,y,t)$ adalah citra hasi penghilangan bayangan dan τ adalah *threshold* intensitas citra bayangan.

Selanjutnya akan dilakukan proses morfologi yang terdiri dari dua tahap yaitu erosi dan dilatasi citra.

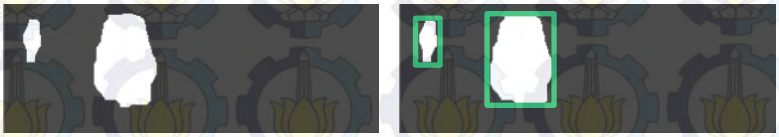


(a) (b)

Gambar 4.7. Citra Hasil Penghilangan Bayangan dan Morfologi. (a) citra hasil *filtering* dan (b) citra hasil penghilangan bayangan dan morfologi.

e. Deteksi objek kendaraan

Pendeteksian objek pada Citra hasil *filtering* dengan analisis *blob* untuk memberi label pada seiap objek yang terdeteksi. Analisis *blob* ini menggunakan metode *connected component*, dimana di setiap kumpulan objek yang tingkat keabuan pikselnya bernilai satu, dikategorikan sebagai satu objek. Dari analisis *blob* tersebut akan diperoleh informasi tentang *centroid*, luas area, tinggi, dan lebar sebuah objek dari bentuk *rectangle*.



(a) (b)

Gambar 4.8. Citra Hasil Deteksi *Blob*, bagian (a) *foreground* dan (b) merupakan *blob* yang terdeteksi.

f. *Tracking* objek kendaraan

Langkah berikutnya adalah *tracking* objek kendaraan dengan *tracking blob*. *Tracking blob* bertujuan untuk menentukan bahwa objek yang terdeteksi pada *frame* ke-*t* dengan *frame* ke-*t+1*

adalah objek yang sama atau berbeda. Hal tersebut dapat diketahui dengan mengukur jarak dari *centroid* masing-masing *blob* yang terdeteksi. Jika jaraknya kurang dari *threshold* yang telah ditentukan maka objek yang terdeteksi pada *frame-frame* tersebut adalah objek yang sama.

g. Proses klasifikasi dan penghitungan kendaraan

Objek hasil *tracking* tersebut akan diklasifikasikan jenisnya menggunakan metode *Geometric Invarian Moment*. Setiap objek hasil *blob tracking* akan dihitung nilai momen invarianya menggunakan persamaan (2.9).

Momen yang akan digunakan untuk mengenali jenis dari objek adalah momen pertama (ϕ_1). Alasan mengenai pemilihan momen pertama (ϕ_1) sebagai parameter klasifikasi dijelaskan pada bab V. Model matematis pengenalan jenis objek kendaraan ditunjukkan pada persamaan (4.2).

$$R(\phi_1) = \begin{cases} motor, \ jika\ \phi_1 \in I \\ mobil, jika\ \phi_1 \in J \\ truk/bus\ , jika\ \phi_1 \in K \end{cases} \qquad (4.2)$$

Dimana $R(\phi_1)$ fungsi pengenalan jenis objek kendaraan. I adalah interval *threshold* momen jenis motor, J adalah interval *threshold* momen jenis mobil, dan K adalah interval *threshold* momen jenis truk/bus.

Pada Gambar 4.9 ditunjukkan hasil klasifikasi jenis kendaraan. Setelah dikenali jenisnya maka tahap selanjutnya adalah penghitungan jumlah dari masing-masing jenis kendaraan.

(a) (b)

Gambar 4.9. Citra Hasil Klasifikasi, bagian (a) citra objek hasil *blob tracking* dan (b) merupakan objek *blob* yang dikenali.

4.1.2 Analisis Kebutuhan Sistem

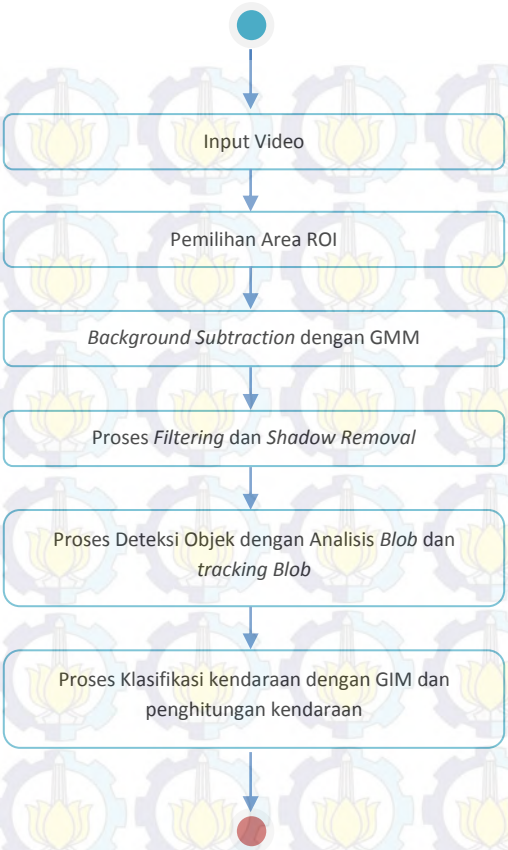
Perangkat lunak ini dibangun menggunakan bahasa pemrograman atau compiler C# dengan IDE Microsoft Visual Studio 2010, dan *frame*work pengolahan citra digital Emgu CV. Emgu CV adalah *wrapper* dari Open CV, dimana Open CV merupakan *library* khusus untuk pengolahan citra digital untuk bahasa C++, Java, dan Phyton. Karena pada Tugas Akhir ini menggunakan bahasa pemrograman C# maka dibutuhkan Emgu CV sebagai penghubung antara Open CV dan C#.

Tabel 4.1. Tabel kebutuhan sistem

No	Perangkat Lunak	Keterangan
1.	IDE Microsoft Visual Studio 2010	Untuk tampilan grafis antarmuka program
2.	Bahasa Pemrograman C#	Untuk proses algoritma dan pengolahan *input* video digital
3.	*Frame*work Emgu CV	*Library computer vision* untuk proses algoritma program

Selain itu juga diperlukan kamera digital untuk merekam arus kendaraan di jalan. Program ini dikembangkan menggunakan laptop dengan spesifikasi processor Intel(R) Core(TM) i3-2328M CPU @2.20 GHz dan RAM sebesar 2 GB. Tabel kebutuhan sistem disajikan pada Tabel 4.1.

Gambar 4.10 di bawah ini menjelaskan tentang gambaran pokok *business rule* dari perangkat lunak yang dibangun.



Gambar 4.10. *Business Rule* Sistem Perangkat Lunak.

4.2 Perancangan Sistem

Setelah analisis sistem, kemudian dilanjutkan dengan perancangan sistem. Perancangan sistem tersebut meliputi perancangan data sistem, perancangan *class* sistem, perancangan proses algoritma sistem, dan perancangan antarmuka sistem.

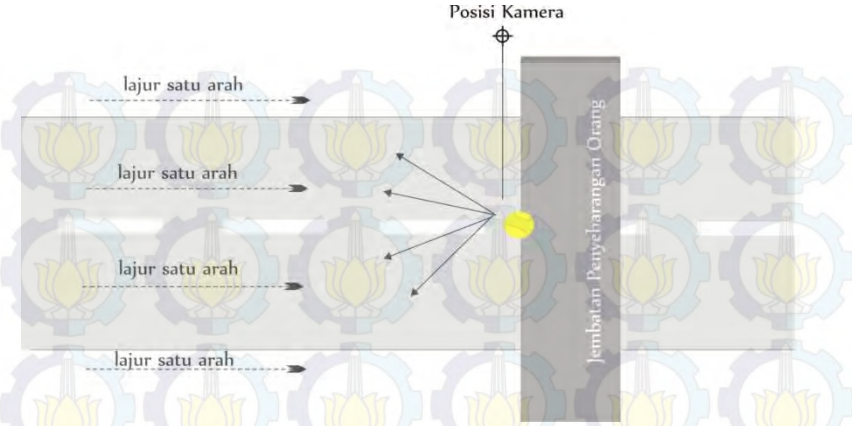
4.2.1 Perancangan Data Sistem

Terdapat tiga macam data yang digunakan oleh sistem ini antara lain data masukan, data proses, dan data keluaran. Pada Tugas Akhir ini data masukan berupa video rekaman *offline* di beberapa ruas jalan diambil secara mandiri oleh peneliti. Data proses merupakan data yang berisi parameter-parameter yang akan digunakan oleh algoritma *Background Subtraction* dan metode *Gaussian Mixture Model* serta algoritma pengklasifikasian jenis kendaraan dengan metode *Geometric Invarian Moment*. Sedangkan data keluaran adalah informasi mengenai jumlah dari setiap jenis kendaraan bergerak yang terdeteksi oleh sistem.

4.2.1.1 Data Masukan

Data masukan sistem ini berupa video *offline* rekaman kendaraan di jalan raya yang diambil menggunakan kamera digital. Video diambil dari jembatan penyeberangan dengan posisi lensa kamera menjangkau ke seluruh area jalan satu lajur. *Layout* untuk pengambilan video rekaman dapat dilihat pada Gambar 4.11. File video yang diambil memiliki spesifikasi sesuai dengan batasan masalah yang terdapat dalam penelitian Tugas Akhir ini yaitu:

a. Memiliki kemampuan untuk merekam video dengan kecepatan minimal 25 *frame per second*.
b. Memiliki kemampuan untuk merekam video dengan resolusi 320 x 240 piksel citra.
c. Format ekstensi rekaman video *offline* dapat berupa: .*avi; .*mov; .*flv; dan .*3gp.



Gambar 4.11. *Layout* Untuk Pengambilan Video Rekaman.
(Sumber: "Penghitungan Kendaraan Bergerak Berbasis Algoritma Background Subtraction Menggunakan Metode Gaussian Mixture Model" [5])

4.2.1.2 Data Proses

Data proses merupakan data yang digunakan dalam proses pengolahan data masukan. Data proses ini diperoleh dari hasil pengolahan data masukan sesuai dengan tahapan algoritma dan metode yang telah disusun. Tabel 4.2 menjelaskan tahapan dari data proses.

Tabel 4.2. Tabel data proses

No	Tahapan	*Input*	*Output*
1.	*Input* Awal	Video	*Frame* Citra
2.	Pilih Area ROI	*Frame* Citra	Citra ROI
3.	Proses GMM	Citra ROI	Citra *Foreground*
4.	Filtering	Citra *Foreground*	Citra *Foreground* yang telah difilter
5.	Deteksi *Blob*	Citra *Foreground* yang telah	Area *blob* (*centroid, width, height*)

		difilter	
6.	Hitung Momen Invarian	Objek hasil deteksi *blob*	Momen Invarian dari setiap objek
7.	Pengklasifikasian	Momen Invarian dari setiap objek	Objek dengan jenisnya
8.	Hitung Kendaraan	Objek dengan jenisnya	Jumlah kendaraan berdasrkan jenisnya.

Sistem ini menggunakan desain parameter yang sama dengan penelitian sebelumnya oleh R. Arif[5] dan terdapat penambahan parameter untuk klasifikasi kendaraan. Tabel di bawah ini mejelaskan desain parameter yang dibutuhkan sistem dalam melakukan proses tahapan pada Tabel 4.2.

Tabel 4.3. Tabel parameter ROI[5]

No.	Variabel	Tipe Data	Keterangan
1.	X	Double	Koordinat titik x pojok kiri area ROI
2.	Y	Double	Koordinat titik y pojok kiri area ROI
3.	kolom	Int	Lebar citra ROI satuan piksel
4.	Baris	Int	Panjang citra ROI satuan piksel

Tabel 4.4. Tabel parameter GMM[5]

No.	Variabel	Tipe Data	Keterangan
1.	nGauss	Int	Panjang komponen Gaussian
2.	lRate	Double	*Learning rate* untuk meng-*update* parameter-parameter gaussian
3.	Shadow	Bool	Untuk mendeteksi bayangan objek

Tabel 4.5. Tabel parameter *filtering* citra[5]

No.	Variabel	Tipe Data	Keterangan
1.	Kernel	Int	Ukuran kernel (*mask*) filter

Tabel 4.6. Tabel parameter deteksi *blob*[5]

No.	Variabel	Tipe Data	Keterangan
1.	minArea	Int	Minimal area luas piksel bernilai 1 yang diidentifikasi sebagai objek
2.	maxArea	Int	Maksimal area luas piksel bernilai 1 yang diidentifikasi sebagai objek
3.	thDistance	Int	Treshold jarak minimal korelasi antara objek *frame* ke-t dengan objek *frame* ke-t-1
4.	thInactive	Int	Treshold maksimal citra hilang dari *track* (waktu satuan *frame*)
5.	thActive	Int	Treshold maksimal citra terdeteksi sebagai objek (waktu satuan *frame*)

Tabel 4.7. Tabel parameter Momen Invarian

No.	Variabel	Tipe Data	Keterangan
1.	Citra Objek	Image	Citra dari objek yang terdeteksi

4.2.1.3 Data Keluaran

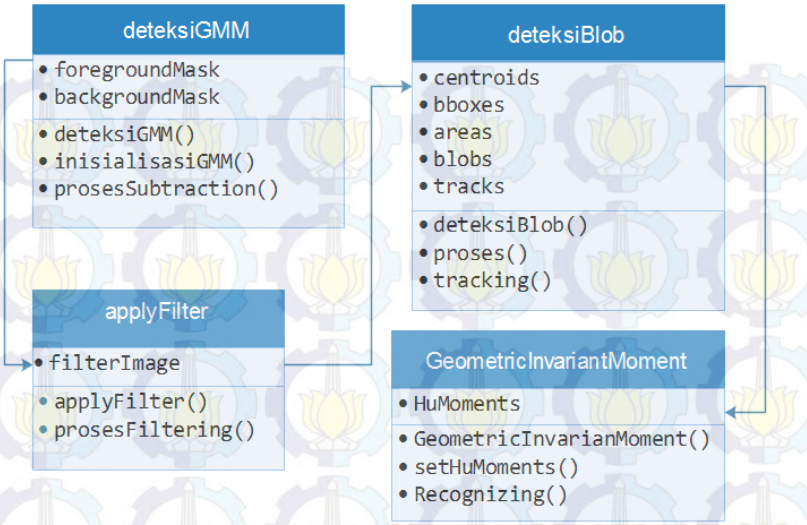
Data keluaran berupa hasil klasifikasi dan penghitungan jumlah kendaraan bergerak. Program ini juga menghasilkan keluaran berupa citra ROI, citra *foreground*, citra *foreground* yang telah difilter, dan citra hasil deteksi *blob*.

4.2.2 Perancangan *Class* Sistem

Dalam membangun sebuah aplikasi diperlukan perancangan *class* sesuai dengan kebutuhan sistem. Pada Tabel 4.8 akan dijelaskan *class* yang ada pada sistem ini dan fungsi dari masing-masing *class* tesebut. Gambar 4.12 menunjukkan digram *class* dari sistem ini.

Tabel 4.8. Tabel *Class* Sistem

No.	Nama *Class*	Fungsi
1	deteksiGMM.cs	Untuk mensegmentasi *frame* menjadi citra *foreground* dan *background.*
2	applyFilter.cs	Menghilangkan *noise* pada citra *foreground* , terdapat beberapa jenis *filter* antara lain Median, Gaussian, Blur, dan Bilatral.
3	deteksiBlob.cs	Mendeteksi objek pada citra *foreground* yang telah difilter dan melakukan *tracking* untuk mengetahui manakah objek yang sama pada *frame* berikutnya.
4	GeometricInvarian Moment.cs	Menghitung HuMoments dari objek, untuk dijadikan acuan pengklasifikasian jenis objek kendaraan.

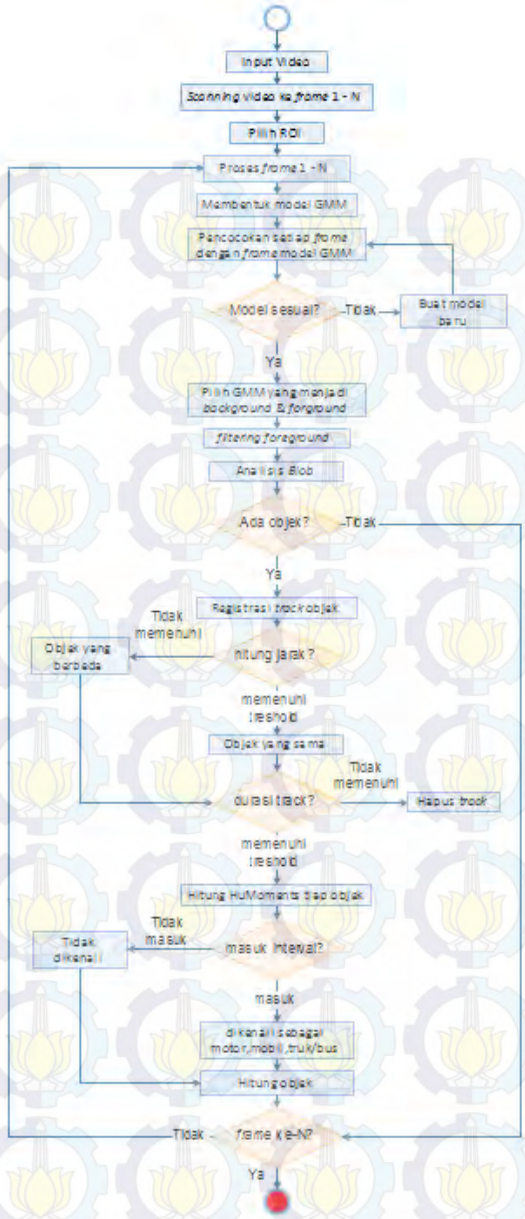


Gambar 4.12. Diagram *Class* Sistem Klasifikasi Kendaraan.

4.2.3 Perancangan Proses Algoritma

Proses algoritma ini dimulai dengan pemilihan ROI (*Region of Intereset*) yang digunakan sebagai area penghitungan dan klasifikasi kendaraan bergerak. Selanjutnya citra ROI tersebut diproses menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model* untuk mengelompokkan bagian piksel yang tergolong citra *foreground* atau *background*[5]. Proses berikutnya adalah *filtering* citra *foreground* untuk menghilangkan *noise*. Citra *foreground* yang telah difilter akan dianalisis *blob* untuk mendeteksi objek yang ada pada citra tersebut. Pada tahap akhir objek-objek yang terdeteksi diklasifikasikan dengan metode *Geometric Invariant Moment* serta dilakukan penghitungan.

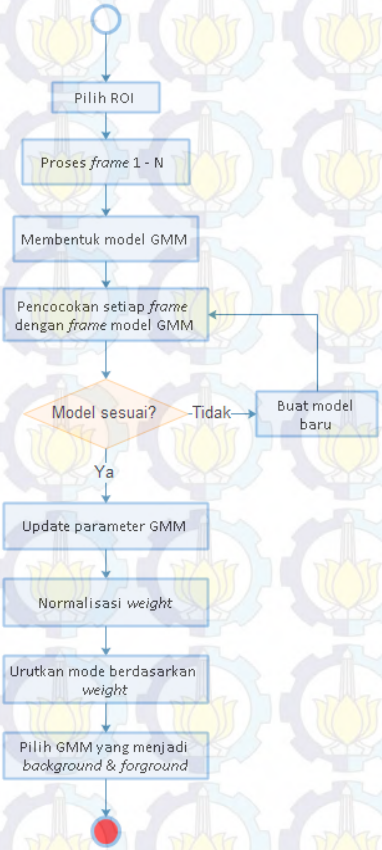
Gambar 4.13 menjelaskan diagram alir sistem perangkat lunak.



Gambar 4.13. Diagram Alir Sistem Perangkat Lunak.

4.2.3.1 Perancangan Proses *Gaussian Mixture Model*

Berikut adalah alur proses untuk mengolah citra per *frame* menggunakan metode GMM. Berdasarkan hasil uji coba *input* parameter GMM pada penelitian R. Arif [5], nilai komponen Gaussian yang baik digunakan adalah 3,4 dan 5, sedangkan nilai *learning rate* adalah 0.001,0.005, dan 0.01. Berikut adalah blok diagram proses GMM.



Gambar 4.14. Diagram Alir GMM.

4.2.3.2 Perancangan Proses Deteksi *Blob*

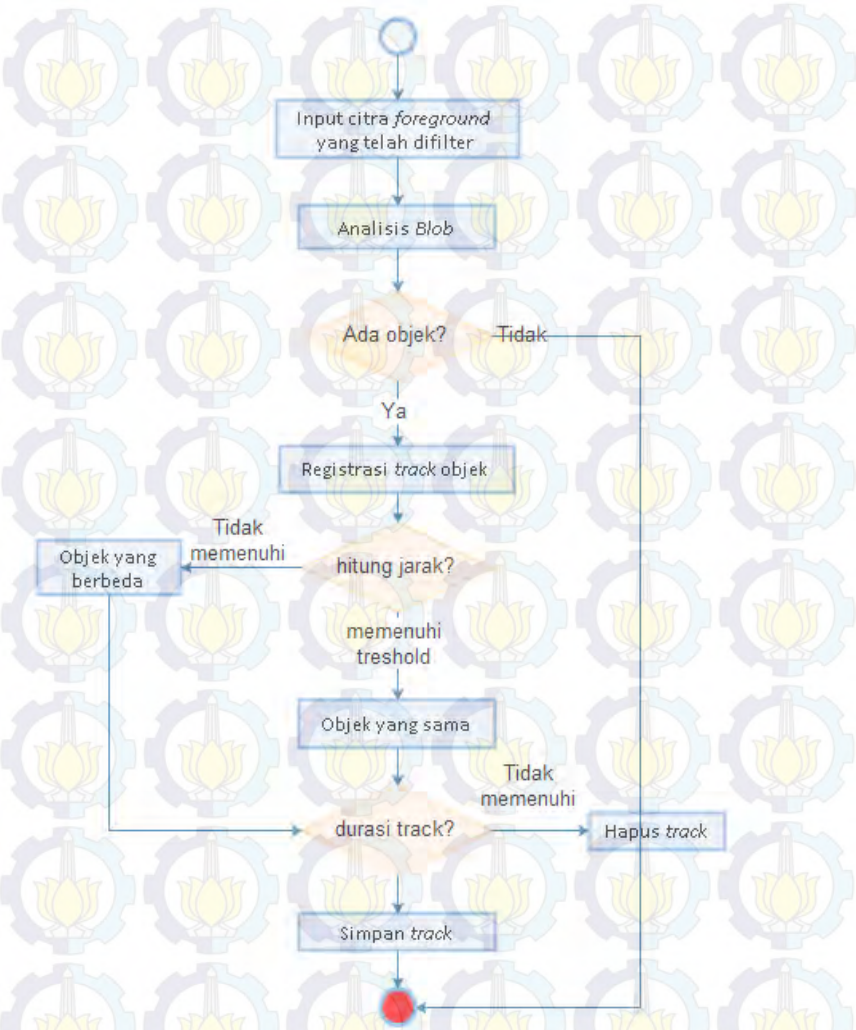
Deteksi *blob* digunakan untuk menentukan apakah objek yang terdeteksi pada *frame* ke-*t-1* sama atau merupakan objek yang berbeda dari objek yang terdeteksi pada *frame* ke-*t.* Analisis *blob* menghasilkan nilai *centroid*, *weight*, dan *height* di setiap objek yang terdeteksi. Dari informasi nilai tersebut, program dapat menghitung jarak antara objek pada *frame* ke-*t-1* dan objek pada *frame* ke-*t*. Apabila jarak yang diketahui dibawah *treshold* yang telah ditentukan, maka objek tersebut termasuk objek yang sama. Jika tidak maka objek tersebut dikategorikan sebagai objek yang lain. Dengan analisis *blob* inilah program dapat menghitung jumlah objek atau kendaraan yang bergerak[5].

Input yang dibutuhkan untuk proses deteksi *blob* adalah:

a. Minimal luas area yang diidentifikasi sebagai objek
b. Maksimal luas area yang diidentifikasi sebagai objek
c. Batas maksimal jarak dua objek di dua *frame* yang berbeda dikategorikan sebagai objek yang sama
d. Batas maksimal *blob* hilang dari *track*
e. Batas maksimal *blob* terdeteksi oleh program dan dihitung sebagai objek

Hasil dari deteksi *blob* ini adalah kumpulan *blob* yang setiap *blob*-nya memiliki informasi *centroid* (titik pusat koordinat *blob*), *weight* (lebar *blob*), dan *height* (panjang *blob*). Informasi tersebut nantinya digunakan untuk proses penghitungan dan klasifikasi jenis kendaraan.

Berikut ini adalah blok diagram proses deteksi *blob*:



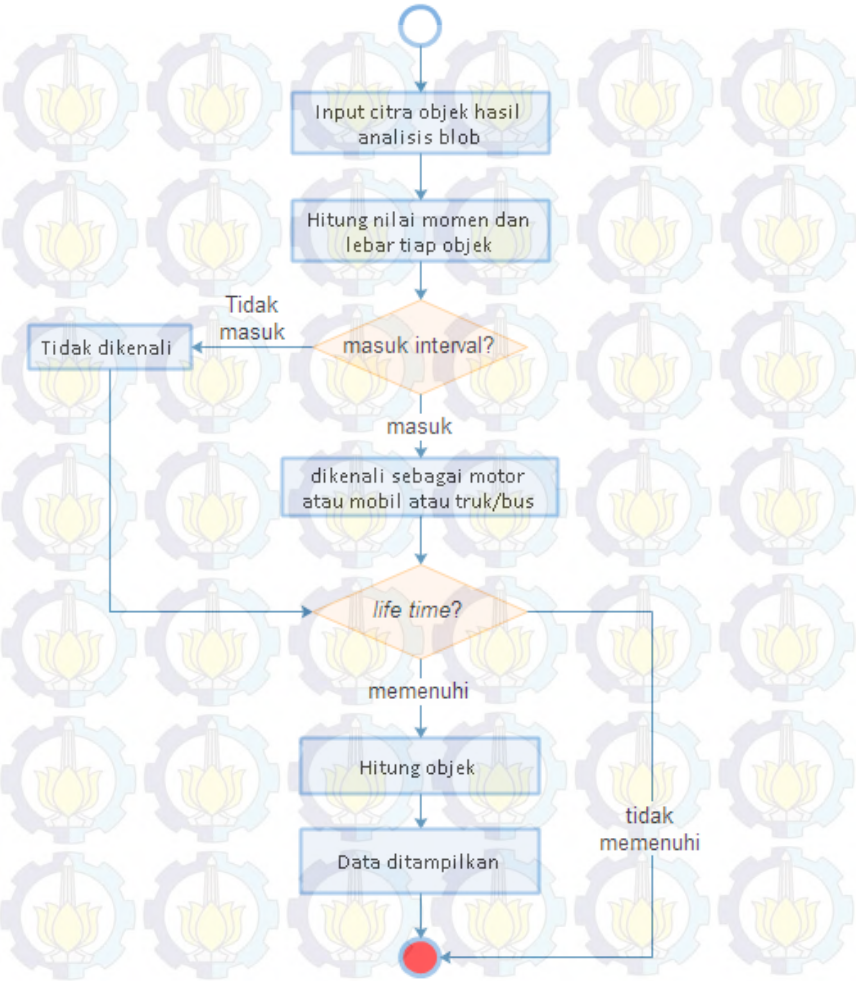
Gambar 4.15. Diagram Alir Deteksi *Blob*.

4.2.3.3 Perancangan Proses Klasifikasi Jenis Kendaraan

Proses klasifikasi jenis kendaraan dengan cara mengolah objek-objek hasil dari analisis *blob*. Objek tersebut akan dihitung nilai momen invarian dan lebarnya untuk dijadikan sebagai parameter klasifikasi. Kemudian menganalisa dari setiap momen invarian dan lebar objek untuk menentukan *threshold* interval dari jenis motor, mobil, dan bus/truk. Analisis parameter klasifikasi dan perumusan algoritma *threshold* akan dijelaskan lebih detail pada Bab V. Setelah dilakukan analisis parameter, diperoleh algoritma klasifikasi dengan nilai masukan berupa nilai parameter klasifikasi yaitu momen invarian pertama (ϕ_1) dan lebar dari objek. Kemudian nilai parameter tersebut akan diperiksa, apabila nilainya masuk dalam *threshold* interval untuk klasifikasi maka objek teridentifikasi sesuai dengan jenis yang didefinisikan pada interval tersebut. Jika nilai parameter tidak masuk dalam interval maka objek tidak dikenali jenisnya.

Proses penghitungan setiap jenis kendaraan dengan cara melihat *life time* dari setiap objek yang terdeteksi pada ROI. *Life time* merupakan masa objek bertahan pada ROI. Jika *life time* dari setiap objek melebihi *threshold* yang ditentukan maka objek akan dihitung sesuai jenisnya. Program ini menggunakan *threshold* untuk *life time* sama dengan 5 *frame*. Kemudian hasil penghitungan akan ditampilkan pada antar muka dari perangkat lunak ini.

Diagram alir klasifikasi menggunakan GIM dapat dilihat pada Gambar 4.16.



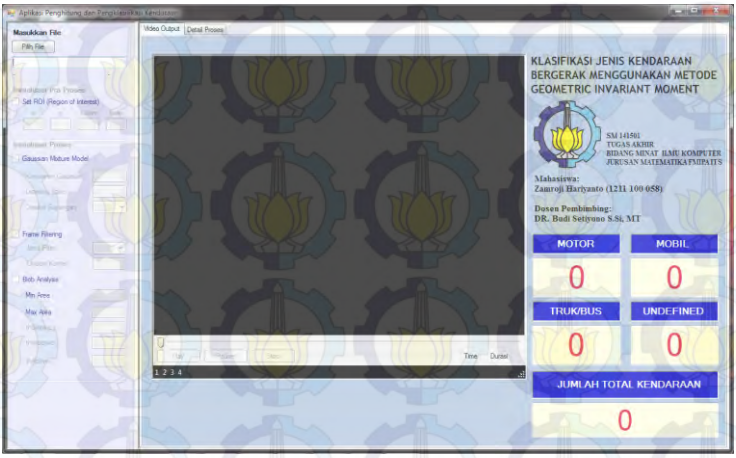
Gambar 4.16. Diagram Alir Klasifikasi Kendaraan Dengan Metode GIM.

4.2.4 Perancangan Antar Muka Sistem

Desain antar muka sistem dibutuhkan agar pengguna dengan mudah mengoperasikan perangkat lunak yang dibangun. Desain ini dibuat semenarik mungkin dan *user friendly*. Berikut adalah desain antar muka halaman utama dan halaman detail proses:

4.2.4.1 Perancangan Halaman Utama

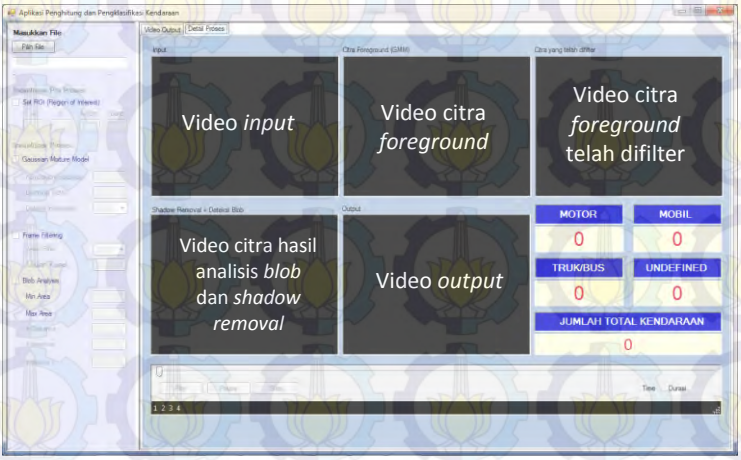
Halaman utama ini memiliki desain antar muka yang ditunjukkan pada Gambar 4.1.7. Pada halaman ini dilakukan proses pemilihan video, pemilihan ROI, dan mengisi parameter proses. Halaman ini menampilkan video hasil klasifikasi jenis kendaraan dan hasil penghitungan setiap jenis kendaraan.



Gambar 4.17. Antar muka halaman awal.

4.2.4.2 Perancangan Halaman Detail

Pada halaman detail ini ditampilkan video *input*, video citra *foreground* hasil GMM, video citra *foreground* yang telah difilter, video citra *foreground* hasil analisis blob dan *shadow removal*, dan video *output* hasil klasifikasi. Desain antar muka dari halaman detail ditunjukkan pada Gambar 4.18.



Gambar 4.18. Antar muka halaman detail.

4.3 Implementasi Sistem

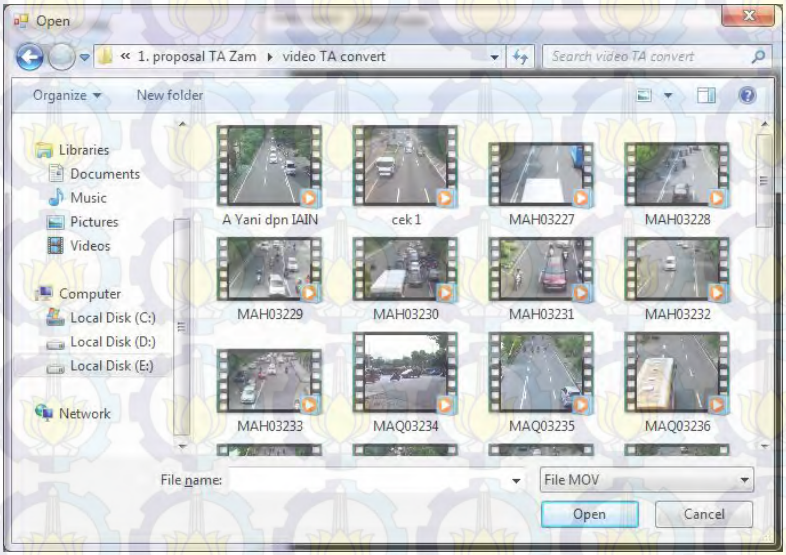
4.3.1 Implementasi *Input* Video

Inputan program berupa video rekaman arus kendaraan dengan tipe, *.avi, *.flv,*.mov, dan *.3gp. Dianjurkan untuk menggunakan video dengan resolusi 240x320 piksel dan memiliki kecepatan 25 fps, supaya proses pengolahan informasi digital tidak berlangsung lama. Semakin tinggi resolusi dan kecepatan video tersebut, maka durasi proses pun semakin lama. Kode program untuk menerima *input* video adalah sebagai berikut.

```
// KETIKA BUTTON OPEN DI CLICK
        private Capture videoInput;
        private int frameRate;
```

```
        private int totalFrame;
        private int baris;
        private int kolom;
        private void button_open_Click(object sender,
EventArgs e)
        {
            if (openFile.ShowDialog() ==
DialogResult.OK)
            {
                if (videoInput != null)
                {
                    if (videoInput.GrabProcessState ==
System.Threading.ThreadState.Running)
                        videoInput.Stop();
                    videoInput.Dispose();
                    pictureBoxMain.Image = null;
                }
                try
                {
                    ........
                    videoInput = new
Capture(openFile.FileName);
                    frameRate =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FPS);
                    totalFrame =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FRAME_COUNT);
                    baris =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FRAME_HEIGHT);
                    kolom =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FRAME_WIDTH);
                    ........
                    //MENGAMBIL TIAP/NEXT FRAME DARI
OBJEK VIDEO
                    videoInput.ImageGrabbed += new
Emgu.CV.Capture.GrabEventHandler(prosesFrame);
                    ........
```

Kode program lebih lengkap dapat dilihat pada lampiran A.1. Gambar 4.19 adalah tampilan antar muka pengambilan *input* video.



Gambar 4.19. Antar muka *input* video

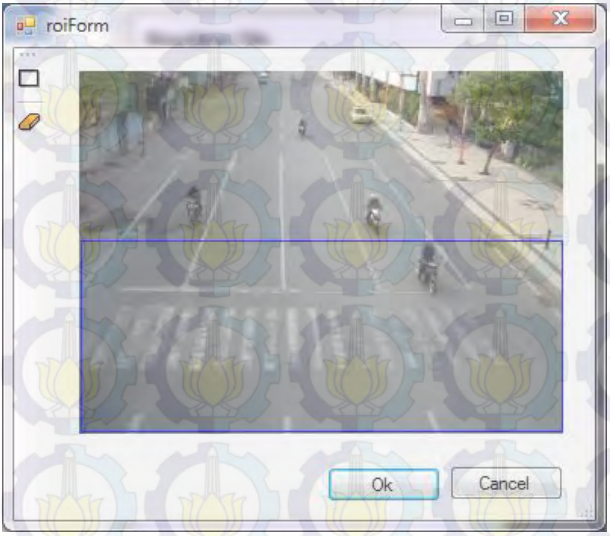
4.3.2 Implementasi Pemilihan area ROI

Region of Interest (ROI) adalah daerah bagian dari citra atau *frame* yang akan diproses pada sistem ini. Pemilihan ROI ini bertujuan untuk memudahkan dan membantu program dalam menspesifikasikan kebutuhan program. Ukuran ROI pada sistem ini dianjurkan memuat setengah dari ukuran *frame* inputan. Sehingga dapat menangkap objek kendaraan secara utuh dan hasil yang diinginkan lebih akurat dan lebih efisien. Hal ini dikarenakan program hanya mengolah informasi dari bagian piksel yang dibutuhkan saja yaitu ROI. Kode program untuk pemilihan ROI adalah sebagai berikut.

```
// EVENT KETIKA CHECKBOX ROI BERUBAH CENTANG
        private Rectangle userDefineRect;
```

```
        private Image<Bgr, byte> frameforSelectedRect;
        private void checkBox_ROI_CheckedChanged(object
sender, EventArgs e)
        {
            if (checkBox_ROI.Checked == true)
            {
                frameforSelectedRect = new Image<Bgr,
byte>(videoInput.QueryFrame().ToBitmap());
                Bitmap frame_1st =
frameforSelectedRect.ToBitmap();
                roiForm form = new roiForm();
                form.firstFrame = frame_1st;
                .....
                .....
        }
```

Kode program selengkapnya disajikan pada Lampiran A.1.Gambar 4.20 adalah tampilan antar muka pemilihan area ROI.



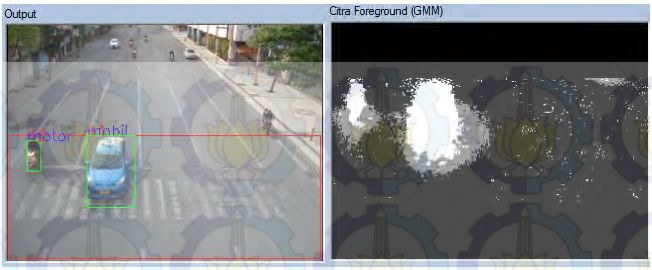
Gambar 4.20. Antar muka pemilihan area ROI

Gambar 4.21 merupakan tampilan program setelah pemilihan ROI dilakukan.

Gambar 4.21. Kotak warna biru pada *frame* tersebut adalah area ROI yang dipilih.

4.3.3 Implementasi Proses GMM

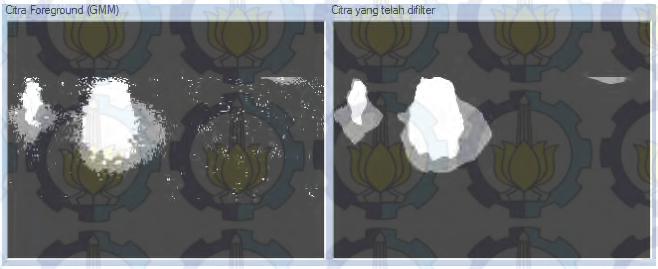
Proses GMM digunakan untuk mencari citra *foreground* dan *background*. Citra *foreground* merupakan citra yang merepresentasikan keadaan objek yang bergerak. Parameter untuk menggunakan proses GMM ini adalah: Banyaknya komponen gaussian digunakan adalah 3,4 dan 5, sedangkan nilai *learning rate* adalah 0.001,0.005, dan 0.01 sesuai hasil penelitian[5]. Kode program class GMM dalam Tugas Akhir ini diambil dari penelitian R. Arif[5] yang dapat dilihat pada Lampiran A.3. Gambar 4.22 menampilkan citra *foreground* hasil pengolahan dari algoritma *Gaussian Mixture Model*.



Gambar 4.22. *Frame* ROI dan citra *foreground* dari citra ROI

4.3.4 Implementasi *filtering* citra

Proses f*iltering* citra dilakukan untuk menghilangkan *noise* atau objek gangguan yang tidak dibutuhkan oleh program. Sehingga program akan dengan mudah mengolah informasi yang ada pada citra Pada program ini disediakan banyak metode filter yang dapat digunakan. Diantaranya adalah; median, Gaussian, blur, dan bilateral. Pada penelitian[5] menyatakan bahwa metode median lebih baik digunakan, karena hasil filter yang mampu menghilangkan *noise* kecil dan memperjelas objek yang terdeteksi. Kode program class *filtering* dalam Tugas Akhir ini diambil dari penelitian R. Arif[5] yang dapat dilihat pada Lampiran A.4. Gambar 4.23 merupakan gambar citra *foreground* yang telah difilter menggunakan metode median.



Gambar 4.23. Citra *foreground* dan citra *foreground* yang telah difilter

Citra yang telah difilter kemudian dilakukan *enhancement* yaitu dengan menghilangkan bayangan dan

melakukan morfologi pada citra. Proses morfologi ini bertujuan untuk menggabungkan daerah yang memiliki intesitas yang sama. Proses ini terdiri dari dua tahap yaitu erosi dan dilatasi. Berikut kode dari proses ini:

```
// CEK KONDISI JIKA FILTERING DI SET OLEH USER
if (checkBox_Filtering.Checked == true)
{
     filter.prosesFiltering(foreground);
     citraTerfilter = filter.filterImage;
     tampilVideo(citraTerfilter.ToBitmap(),
pictureBox3);
     fgRemovShad =
citraTerfilter.ThresholdToZero(tresh).Erode(4).Dilate(3
);
     tampilVideo(fgRemovShad.ToBitmap(), pictureBox4);
}
     .....
```

Kode lebih lengkap pada Lampiran A.1. Citra hasil proses penghilangan bayangan dan morfologi dapat dilihat pada Gambar 4.24.

Gambar 4.24. Citra *foreground* hasil *filtering* dan citra *foreground* yang telah dilakukan proses penghilangan bayangan dan morfologi

4.3.5 Implementasi Deteksi *blob*

Deteksi *blob* digunakan untuk menemukan kumpulan *blob* pada suatu citra. Dimana *blob-blob* tersebut akan disimpan sebagai objek kendaraan yang kemudian diolah lebih lanjut untuk mengklasifikasikan jenis dan menghitung jumlah kendaraan bergerak. Parameter untuk menggunakan deteksi ini sesuai dengan Tabel 4.6. Untuk proses deteksi *blob* dibuat *class* deteksi *blob* yang memiliki beberapa properti dan fungsi. Kode program class deteksi *blob* dalam Tugas Akhir ini diambil dari penelitian R. Arif[5] yang dapat dilihat pada Lampiran A.5.

4.3.6 Implementasi Pengklasifikasian dan Penghitungan Kendaraan

Implementasi ini merupakan tahap terakhir dan juga merupakan fungsi utama program. Implementasi ini dibagi menjadi dua yaitu implementasi pengklasifikasian jenis kendaraan dan penghitungan kendaraan.

Objek-objek hasil dari analisis blob akan diidentifikasi jenisnya menggunakan GIM, yang selanjutnya dihitung berdasarkan jenisnya. Alur program sesuai dengan rancangan pada sub bab 4.2.4.3. Untuk aplikasi ini dibutuhkan *class Geometric Invariant Moment*, berikut ini kodenya:

```
namespace VehicleCounting
{
    class GeometricInvarianMoment
    {
        //Field
        private Image<Gray, Byte> gray_image;
        private MCvMoments Moments;
        private MCvHuMoments HuMoments;
        private double[] HuM = new double[7];
        int id = 8, idmin, area;
        double sigma;
        double[] d = new double[100];
        double[,] GIMtarget;
        string[] identitas;
        //konstruktor
        public GeometricInvarianMoment() {}
```

```
        public GeometricInvarianMoment(Image<Gray,
Byte> grayImage)
        {
            this.gray_image = grayImage;
            this.Moments = grayImage.GetMoments(true);
            this.HuMoments = Moments.GetHuMoment();
            this.setHuMoments(HuMoments);
        }

        //destruktor
        ~GeometricInvarianMoment() { }

        // Methods
        public void setHuMoments(MCvHuMoments Hu)
        {
            this.HuM[0] = Hu.hu1;
            this.HuM[1] = Hu.hu2;
            this.HuM[2] = Hu.hu3;
            this.HuM[3] = Hu.hu4;
            this.HuM[4] = Hu.hu5;
            this.HuM[5] = Hu.hu6;
            this.HuM[6] = Hu.hu7;
        }
        public double[] getHuMoments()
        {
            return this.HuM;
        }

        public string Recognizing()
        {
            if (this.getHuMoments()[0] >= 0.174 &&
this.getHuMoments()[0] <= 1)
            {
                if (this.gray_image.Width <= 30)
                {
                    return "motor";
                }
                else if (this.gray_image.Width > 30 &&
this.gray_image.Width <= 92)
                {
```

```
                    return "mobil";
                }
                else
                {
                    return "truk";
                }
            }
            else if (this.getHuMoments()[0] >= 0.16 &&
this.getHuMoments()[0] < 0.174)
            {
                if (this.gray_image.Width <= 30)
                {
                    return "motor";
                }
                else if (this.gray_image.Width > 30 &&
this.gray_image.Width <= 92)
                {
                    return "mobil";
                }
                else
                {
                    return "truk";
                }
            }
            else
            {
                return "";
            }
        }
    }
}
```

Kode selengkapnya pada Lampiran A.6. Selanjutnya setiap objek yang telah diidentifikasi jenisnya akan dihitung kemudian hasil hitungannya ditampilkan ke antar muka. Berikut ini kode program pengklasifikasian jenis kendaraan.

```
// EVENT UNTUK PROSES TIAP FRAME KETIKA BUTTON PLAY DI
KLIK
.....
.....
// KLASIFIKASI JENIS KENDARAAN DENGAN GIM
```

```
                    foreach (KeyValuePair<UInt32,
CvTrack> tr in deteksiblob.tracks)
                    {
                        gray_image =
frameROI.Convert<Gray,
Byte>().Copy(tr.Value.BoundingBox);

                        GIM = new
GeometricInvarianMoment(gray_image);
                        PointF koord = new
PointF((float)tr.Value.BoundingBox.X +
userDefineRect.X, (float)tr.Value.BoundingBox.Y +
userDefineRect.Y);
// MEMBERI LABEL JENIS KENDARAAN PADA OBJEK
frameAwal.Draw(GIM.Recognizing(), ref font,
Point.Round(koord), new Bgr(255.0, 0.0, 0.0));
                        if (tr.Value.BoundingBox.Y != 0
&& tr.Value.BoundingBox.Y != userDefineRect.Height &&
tr.Value.Lifetime >6 &&
!isExistIdObjek(tr.Value.Id.ToString()))
                        {

ObjekList.Add(tr.Value.Id.ToString());

updateCounting(GIM.Recognizing());
                                idShow = jumMobil +
jumMotor + jumTruk + jumUndefined;

                        }
                    }
.....
.....
// FUNGSI UNTUK UPDATE PENGHITUNGAN JENIS KENDARAAN
public void updateCounting(string jenis)
        {
            if (jenis == "motor")
            {
                jumMotor++;
                txtMotor.Text = jumMotor.ToString();
                txtMotor1.Text = jumMotor.ToString();
```

```
            }
            else if (jenis == "mobil")
            {
                jumMobil++;
                txtMobil.Text = jumMobil.ToString();
                txtMobil1.Text = jumMobil.ToString();
            }
            else if (jenis == "truk")
            {
                jumTruk++;
                txtTruk.Text = jumTruk.ToString();
                txtTruk1.Text = jumTruk.ToString();
            }
            else if (jenis == "")
            {
                jumUndefined++;
                txtUndefined.Text =
jumUndefined.ToString();
                txtUndefined1.Text =
jumUndefined.ToString();
            }
        }

        public bool isExistIdObjek(string id)
        {
            bool exist=false;
            foreach (string objek in ObjekList)
            {
                if (objek == id)
                    exist = true;
            }
            return exist;
        }
.....
.....
```

Kode selengkapnya terdapat pada Lampiran A.1. Gambar 4.25 adalah antar muka dari hasil klasifikasi jenis kendaraan.

Gambar 4.25. Antar muka aplikasi saat pengklasifikasian dan penghitungan jenis kendaraan.

Gambar 4.26 menampilkan detail proses dari aplikasi. Halaman ini menampilkan citra dengan area ROI, citra *foreground*, citra *filtering foreground*, citra hasil deteksi *blob* dan penghitungan kendaraan bergerak.

Gambar 4.26. Detail Proses Aplikasi.

BAB V
PENGUJIAN DAN PEMBAHASAN

Bab ini menjelaskan mengenai pengujian program dan pembahasan dari hasil uji coba. Tahap awal yang akan dilakukan adalah meneNtukan parameter dan *threshold* untuk proses klasifikasi jenis kendaraan. Dalam penentuan parameter dan *threshold,* akan dilakukan uji coba terhadap dua video yang memiliki durasi 1 menit. Sehingga akan diperoleh parameter dan *threshold default* yang dapat digunakan sistem untuk klasifikasi jenis kendaraan.

Pengujian tahap berikutnya adalah, pengujian program dengan *input* video rekaman arus lalu lintas di Jalan Embong Malang dengan durasi 3 menit. Pengujian dilakukan dengan melakukan rekaman sebanyak dua kali pada waktu pagi dan siang hari. Waktu pagi hari merepresentasikan kondisi kendaraan di jalan relatif sepi dan tanpa bayangan, sedangkan siang hari merepresentasikan kondisi kendaraan di jalan relatif padat dan memiliki bayangan.

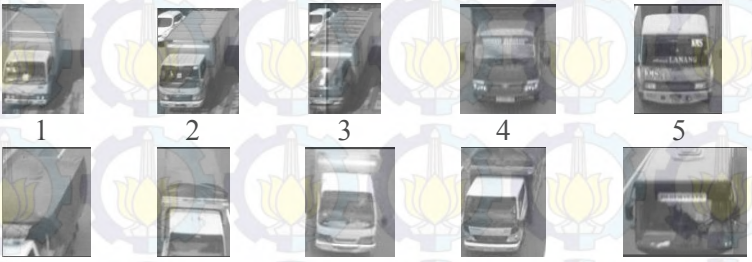
5.1 Analisis dan Uji Coba Parameter GIM.

Untuk memperoleh *threshold* yang sesuai, diperlukan perhitungan nilai momen invarian dari setiap jenis kendaraan. Jenis kendaraan yang akan dikenali dalam penelitian ini antara lain: motor, mobil, dan truk/bus. Dalam penelitian ini dipilih 10 citra dengan objek kendaraan dari masing-masing jenis kendaraan untuk dianalisa. Gambar dari setiap citra dapat dilihat pada Gambar 5.1, 5.2, dan 5.3.

1 2 3 4 5 6 7 8 9 10

Gambar 5.1. 10 Citra Dengan Objek Kendaraan Berjenis Motor.

Gambar 5.2. 10 Citra Dengan Objek Kendaraan Berjenis Mobil.



Gambar 5.3. 10 Citra Dengan Objek Kendaraan Berjenis Truk/Bus.

Nilai momen invarian dari masing-masing jenis kendaraan ditunjukkan pada Tabel 1, 2 dan 3.

Tabel 5.1. Lebar dan Momen Invarian dari 10 Citra Objek Motor.

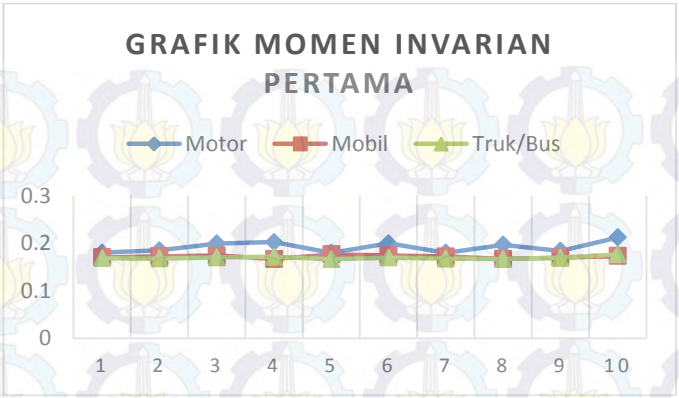
Objek	Lebar	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
1.	22	0,18032	0,00482	0	0	0	0	0
2.	20	0,18515	0,00660	0	0	0	0	0
3.	21	0,19943	0,01207	0	0	0	0	0
4.	12	0,20259	0,01351	0	0	0	0	0
5.	14	0,17998	0,00482	0	0	0	0	0
6.	12	0,19980	0,00482	0	0	0	0	0
7.	15	0,17998	0,01230	0	0	0	0	0
8.	16	0,19665	0,00110	0	0	0	0	0
9.	19	0,18406	0,00620	0	0	0	0	0
10.	15	0,21218	0,01739	0	0	0	0	0

Tabel 5.2. Lebar dan Momen Invarian dari 10 Citra Objek Mobil.

Objek	Lebar	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
1.	45	0,17060	0,0013	0	0	0	0	0
2.	42	0,17269	0,0020	0	0	0	0	0
3.	85	0,17414	0,0025	0	0	0	0	0
4.	59	0,16698	0,0001	0	0	0	0	0
5.	39	0,17653	0,0034	0	0	0	0	0
6.	41	0,17476	0,0027	0	0	0	0	0
7.	34	0,17212	0,0018	0	0	0	0	0
8.	37	0,16706	0,0001	0	0	0	0	0
9.	37	0,16976	0,0010	0	0	0	0	0
10.	43	0,17326	0,0022	0	0	0	0	0

Tabel 5.3. Lebar dan Momen Invarian dari 10 Citra Objek Truk/Bus.

Objek	Lebar	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
1.	103	0,168393	0,00058	0	0	0	0	0
2.	90	0,167955	0,00043	0	0	0	0	0
3.	102	0,170107	0,00116	0	0	0	0	0
4.	149	0,171552	0,00165	0	0	0	0	0
5.	121	0,166677	7,4E-06	0	0	0	0	0
6.	103	0,169786	0,00105	0	0	0	0	0
7.	114	0,167511	0,00028	0	0	0	0	0
8.	107	0,166772	4,03E-05	0	0	0	0	0
9.	92	0,169596	0,00099	0	0	0	0	0
10.	179	0,176571	0,00340	0	0	0	0	0

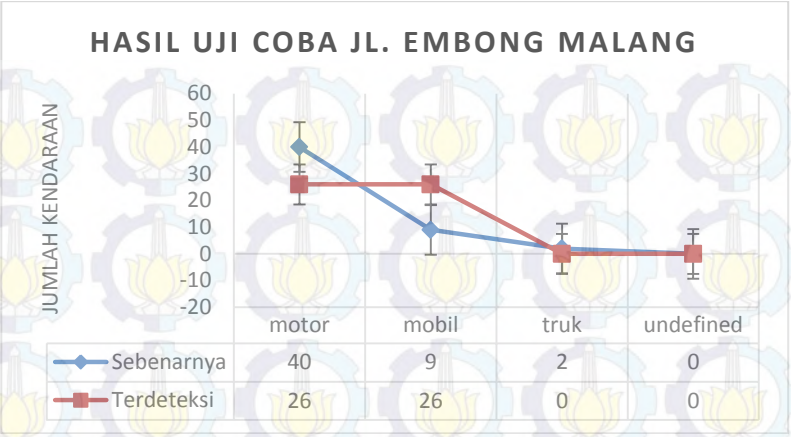


Gambar 5.4. Grafik nilai momen invarian ϕ_1 dari setiap jenis objek.

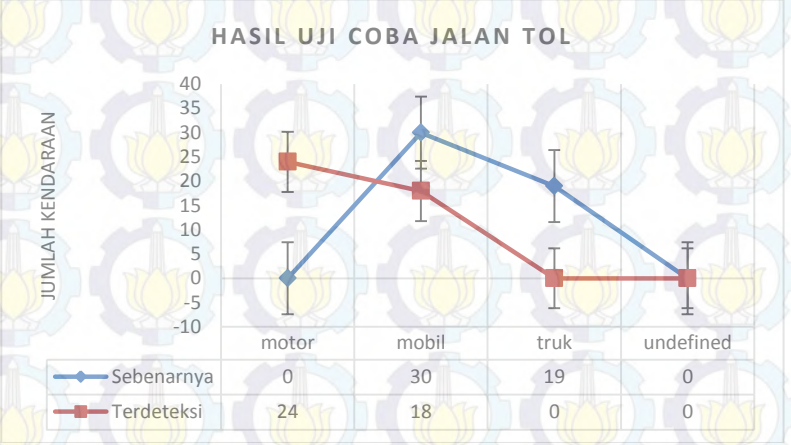
Dari Tabel 2, 3, dan 4 dapat dilihat bahwa nilai momen invarian ϕ_1 lebih signifikan dibandingkan dengan nilai momen invarian yang lain, sehingga momen ϕ_1 akan dijadikan *threshold* untuk klasifikasi jenis kendaraan. Dari grafik pada pada Gambar 5.4 terlihat nilai momen ϕ_1 dari jenis mobil memiliki interval sendiri dan momen ϕ_1 dari jenis mobil dan truk/bus masuk dalam interval yang sama. Sehingga diperoleh interval momen ϕ_1 0,174 s.d 1 untuk jenis motor dan sedangkan jenis mobil dan truk/bus memiliki interval yang sama yaitu 0,16 s.d 0,174. Kemudian dirumuskan algoritma *threshold* (1) sebagai berikut:

if $0{,}174 \leq \phi_1 \leq 1$ *then MOTOR*
else if $0.16 \leq \phi_1 < 0{,}174$ *then MOBIL*
else then UNDEFINED

Selanjutnya dilakukan pengujian dengan masukan rekaman video di Jalan Embong Malang dan di Jalan Tol dengan durasi 1 menit. Penghitungan dimulai dari *frame* ke-125 dikarenakan proses GMM membutuhkan beberapa *frame* untuk penentuan citra *foreground* dan *background*.



Gambar 5.5. Grafik Pengujian Algoritma *Threshold* (1) Pada Video di Jl. Embong Malang.



Gambar 5.6. Grafik Pengujian Algoritma *Threshold* (1) Pada Video di Jalan Tol.

Dari Gambar 5.4 dan 5.5 terlihat bahwa terjadi kesalahan klasifikasi. Hal tersebut dikarenakan interval jenis mobil dan truk/bus sama sehingga diperlukan parameter tambahan untuk

memisahkan jenis tersebut. Selain itu kesalahan identikasi juga dikarenakan beberapa faktor. Faktor tersebut antara lain:

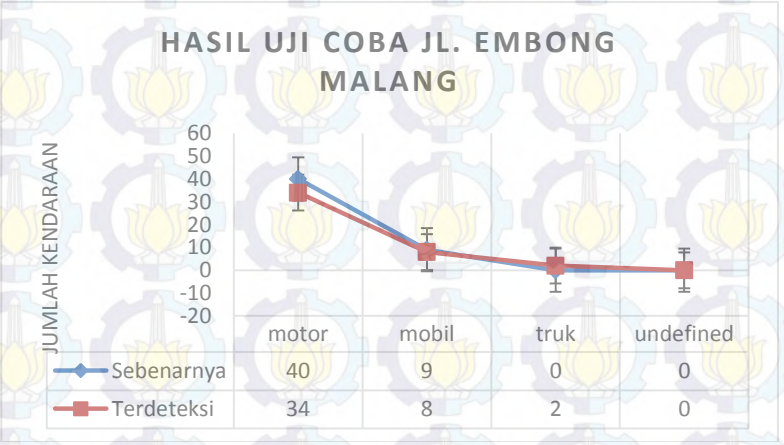
1. Warna objek mirip dengan warna *background* sehingga mengakibatkan ukuran objek yang terdeteksi lebih kecil dari ukuran sebenarnya atau malah tidak terdeteksi.
2. Kontras dan *brightness* pada citra yang mengakibatkan ukuran objek menjadi lebih besar dari ukuran sebenarnya.
3. Bayangan dari objek yang tidak hilang sempurna sehingga ukuran objek menjadi lebih besar dari ukuran sebenarnya.

Gambar 5.7. Faktor-Faktor Pembuat Kesalahan Klasifikasi. Bagian (a) objek motor yang terdeteksi memiliki ukuran yang lebih kecil sehingga nilai momennya mirip momen mobil, (b) objek mobil yang yang terdeteksi memiliki ukuran yang lebih besar sehingga nilai momennya mirip momen truk, dan (c) objek mobil yang yang terdeteksi sebagai motor dikarenakan memiliki momen yang mirip dengan motor.

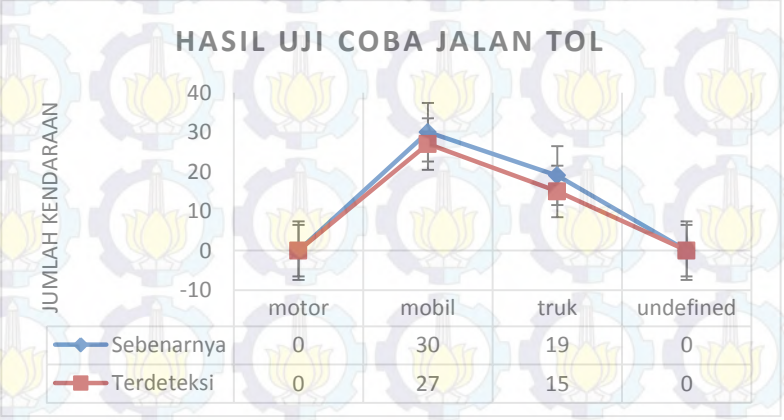
Maka dari itu perlu ditambahkan satu parameter lagi untuk meningkatkan akurasi pengklasifikasian dan untuk memisahkan jenis kendaraan mobil dan truk/bus yaitu berupa lebar objek dikarenakan lebar objek memiliki interval yang signifikan. Dari Tabel 1, 2 dan 3 dapat diperoleh interval lebar objek; 10 s.d 30 untuk jenis motor, 31 s.d 92 untuk jenis mobil, dan lebih dari 92 untuk jenis truk/bus. Dengan mengkombinasikan interval dari nilai momen ϕ_1 dan ukuruan objek citra kendaraan, dirumuskan algoritma *threshold* (2) sebagai berikut:

if $0{,}174 \leq \phi_1 \leq 1$ *then*
  if lebar ≤ 30 *then MOTOR*
  else if $30 <$ *lebar* ≤ 92 *then MOBIL*
  else then TRUK/BUS

else if $0.16 \leq \phi_1 < 0{,}174$ *then*
if lebar ≤ 30 *then MOTOR*
else if $30 <$ *lebar* ≤ 92 *then MOBIL*
else then TRUK/BUS
else then UNDEFINED



Gambar 5.8. Grafik Pengujian Algoritma *Threshold* (2) Pada Video di Jl. Embong Malang.



Gambar 5.9. Grafik Pengujian Algoritma *Threshold* (2) Pada Video di Jalan Tol.

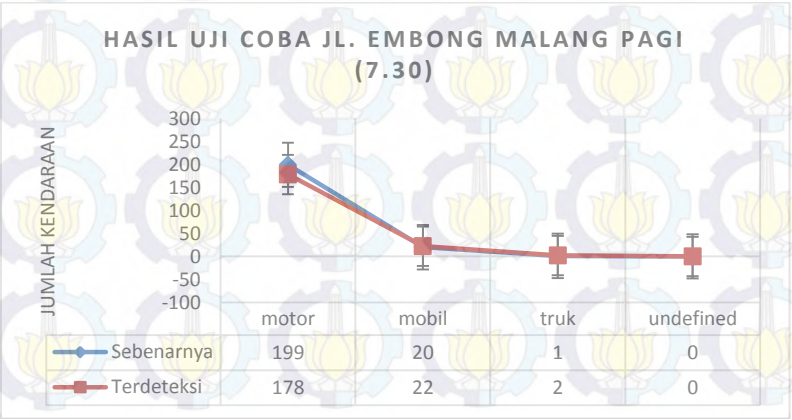
Dari Gambar 5.8 dan 5.9 terlihat bahwa hasil pengujian menggunakan algoritama *threshold* (2) menghasilkan *error* yang kecil. Sehingga diputuskan untuk menjadikan algoritma *threshold* (2) menjadi *default* pada sistem ini.

5.2 Uji Coba Klasifikasi Kendaraan Bergerak

5.2.1 Pengujian Klasifikasi Kendaraan

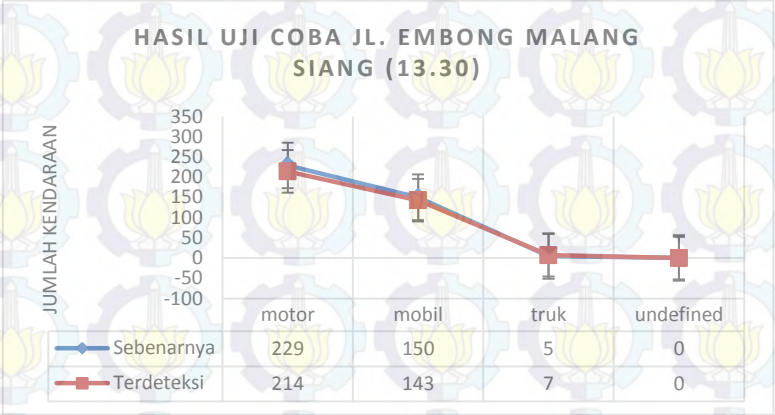
Untuk uji coba dalam penelitian ini menggunakan rekaman video arus kendaraan di Jalan Embong Malang saat pagi dan saat siang yang berdurasi 3 menit. Dengan spesifikasi sebagai video memiliki kecepatan minimal 25 *frame per second*, dengan resolusi 320 x 240 piksel citra, dan format ekstensi rekaman video berupa: .*avi; .*mov; .*flv; atau .*3gp.

Untuk parameter nilai GMM digunakan pada penelitian [5] yaitu parameter komponen Gaussian dengan nilai 4, sedangkan nilai *learning rate* adalah 0.001. Penghitungan dimulai dari *frame* ke-125 dikarenakan proses GMM membutuhkan beberapa *frame* untuk penentuan citra *foreground* dan *background* serta digunakan algoritma *threshold* (2) yang diperoleh pada sub bab 5.1. Hasil dari pengujian klasifikasi jenis kendaraan dapat dilihat pada Gambar 5.10 dan 5.11.



Gambar 5.10. Grafik Klasifikasi Kendaraan Pagi Hari.

Analisis pengujian klasifikasi kendaraan pagi hari pada Gambar 5.10, kesalahan klasifikasi disebabkan terdapat kendaraan yang berjalan berhimpitan sehingga dianggap menjadi satu objek, sehingga ciri objek menjadi berubah mengakibatkan kesalahan klasifikasi. Selain itu juga terdapat objek yang tidak terdeteksi dikarenakan memiliki warna yang mirip dengan *background* sehingga objek tidak terklasifikasi. Juga terdapat kendaraan motor yang membawa banyak barang sehingga dikenali sebagai mobil.



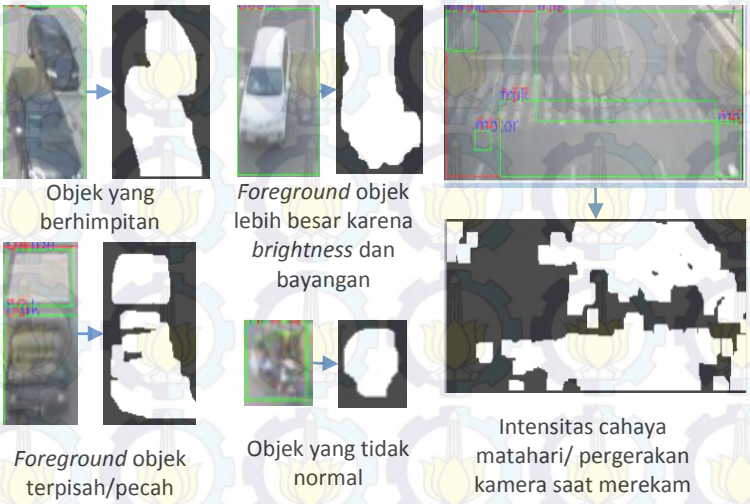
Gambar 5.11. Grafik Klasifikasi Kendaraan Siang Hari.

Analisis pengujian klasifikasi kendaraan pagi siang pada Gambar 5.11, kesalahan klasifikasi disebabkan hal yang sama dengan klasifikasi pada pagi hari. Kesalahan klasifikasi yang menonjol dikarenakan bayangan kendaraan yang tidak hilang sempurna. Selain itu perubaha intensitas cahaya matahari yang terjadi secara tiba-tiba mengakibatkan kesalahan klasifikasi karena deteksi objek salah.

5.2.2 Pembahasan Uji Coba Penghitungan Kendaraan Bergerak

Kemampuan program dalam mengklasifikasikan jenis kendaraan bergerak bergantung kepada beberapa faktor, antara lain:

- Terdapat beberapa objek yang berhimpitan sehingga dianggap satu objek.
- *Foreground* yang lebih besar akibat *brightness* dan bayangan dari objek.
- Intensitas cahaya matahari dan pergerakan kamera saat merekam mengakibatkan penyesuain *background* dan *foreground* lagi dan muncul *noise* yang dianggap objek.
- Objek yang tidak normal seperti motor yang mengangkut banyak barang dan truk yang mengangkut kendaraan lain.
- *Foreground* yang terdeteksi terpisah/pecah mengakibatkan objek dikenali menjadi lebih dari satu objek hal ini senring terjadi pada objek kendaraan truk sehingga mengakibatkan klasifikasi jenis truk/bus memiliki akurasi yang kecil.



Gambar 5.12. Gambar Faktor-Faktor Yang Mempengaruhi Klasifikasi.

Dilihat dari grafik pada Gambar 5.10 dan 5.11. Program dapat mengklasifikasikan jenis kendaraan bergerak dengan baik pada waktu pagi hari dan siang hari. Kemudian akan dilakukan penghitungan tingkat keberhasilan program dalam mengklasifikasikan jenis kendaraan. Penghitungan tingkat akurasi keberhasilan ini berdasarkan pada:

$$PK = \frac{JP}{JS} \times 100\%$$

dengan,
PK = Prosentase Keberhasilan,
JP = Jumlah kendaraan yang terdeteksi dengan benar
JS = Jumlah kendaraan sebenarnya

Maka, tingkat akurasi keberhasilan program dalam menghitung kendaraan bergerak dapat dilihat pada tabel berikut:

a. Jalan Embong Malang Pagi Hari

Tabel 5.4. Prosentase Keberhasilan di Jl. Embong Malang Pagi

	Motor	Mobil	Truk/ Bus	Undefined	Total
Nyata	199	20	1	0	220
Terdeteksi	178	22	2	0	202
Teridentifikasi dengan benar			199		
Akurasi			90,45%		91,81%

b. Jalan Embong Malang Siang Hari

Tabel 5.5. Prosentase Keberhasilan di Jl. Embong Malang Siang

	Motor	Mobil	Truk/ Bus	Undefined	Total
Nyata	229	150	5	0	384
Terdeteksi	214	143	7	0	364
Teridentifikasi dengan benar			362		
Akurasi			94,27%		94,79%

Dari data prosentase keberhasilan pada tabel 5.4 dan 5.5, program dapat mengklasifikasikan jenis kendaraan bergerak baik pada waktu pagi hari maupun siang hari. Program mampu mengklasifikasikan jenis kendaraan dengan tingkat akurasi tertinggi 94,27%.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan. Di samping itu, pada bab ini juga dimasukkan beberapa saran yang dapat digunakan jika penelitian ini ingin dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian yang telah dilakukan terhadap sistem klasifikasi jenis kendaraan berbasis *Geometric Invariant Moment*, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Tugas Akhir ini telah berhasil melakukan klasifikasi jenis kendaraan dengan tahapan: pemilihan ROI, segmentasi citra dengan metode GMM, proses filtering, deteksi *blob* dan *tracking*, pengklasifikasian kendaraan dengan *Geometric Invariant Moment* dan tahap penghitungan kendaraan.
2. Parameter klasifikasi yang digunakan adalah nilai momen pertama (ϕ_1) dan penambahan parameter lebar objek untuk meningkatkan akurasi dan menghindari kesalahan pengenalan akibat kemiripan warna citra objek dengan *background,* kontras dan *brightness* pada citra objek, dan bayangan tidak hilang sempurna.
3. *Geometric Invariant Moment* pada Tugas Akhir ini dapat digunakan untuk mengklasifikasikan kendaraan dalam 3 jenis kendaraan yaitu motor, mobil, dan truk/bus dengan tingkat akurasi tertinggi sebesar 94,27%.

6.2 Saran

Dengan melihat hasil yang dicapai pada penelitian ini, ada beberapa hal yang penulis sarankan untuk pengembangan selanjutnya yaitu:

1. Program belum dapat membedakan jenis bus dan truk. Sehingga untuk penelitian berikutnya disarankan untuk menambahkan jenis klasifikasi bus dan truk.
2. Program belum dapat menghilangkan bayangan objek secara sempurna sehingga mempengaruhi proses klasifikasi. Untuk penelitian berikutnya diharapkan untuk mengkhususkan pada proses penghilangan bayangan objek.
3. Untuk penelitian selanjutnya disarankan untuk menggunakan metode *shape based* atau 3-D *model based*.
4. Program belum terhubung dengan database. Sehingga data penghitungan hanya disimpan sementara oleh memori CPU. Pada penelitian berikutnya program dapat dihubungkan dengan database. Sehingga data dapat disimpan dan diolah untuk kepentingan lebih lanjut.

LAMPIRAN A

A.1 Kode Program Class mainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;

using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Emgu.CV.VideoSurveillance;
using Emgu.CV.Cvb;


using System.Threading;

namespace VehicleCounting
{
    public partial class mainForm : Form
    {
        dataKendaraan dk = new dataKendaraan();
        public mainForm()
        {
            InitializeComponent();
            Control.CheckForIllegalCrossThreadCalls =
false;
            //this.setGIMtarget();
        }

        // KETIKA BUTTON OPEN DI CLICK
        private Capture videoInput;
        private int frameRate;
        private int totalFrame;
```

LAMPIRAN A (LANJUTAN)

```
        private int baris;
        private int kolom;
        private int idTracking;
        private void button_open_Click(object sender,
EventArgs e)
        {
            if (openFile.ShowDialog() ==
DialogResult.OK)
            {
                if (videoInput != null)
                {
                    if (videoInput.GrabProcessState ==
System.Threading.ThreadState.Running)
                        videoInput.Stop();
                    videoInput.Dispose();
                    pictureBoxMain.Image = null;
                }
                try
                {
                    //ENABLED BUTTON
                    buttonPlay.Enabled = true;
                    buttonPause.Enabled = true;
                    buttonStop.Enabled = true;
                    buttonPlay2.Enabled = true;
                    buttonPause2.Enabled = true;
                    buttonStop2.Enabled = true;
                    checkBox_ROI.Checked = false;
                    checkBox_GMM.Checked = false;
                    checkBox_Filtering.Checked = false;
                    checkBox_Blob.Checked = false;
                    resetCounting();
                    //INISIALISASI FILE INPUT BESERTA
AKSES PROPERTI OBJEK
                    videoInput = new
Capture(openFile.FileName);
                    frameRate =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FPS);
```

LAMPIRAN A (LANJUTAN)

```
                    totalFrame =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FRAME_COUNT);
                    baris =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FRAME_HEIGHT);
                    kolom =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_FRAME_WIDTH);
                    //TAMPILKAN DURASI VIDEO
                    double durasi = totalFrame /
frameRate;
                    labelDurasi.Text = "/ " +
TimeSpan.FromSeconds(Math.Floor(durasi)).ToString();
                    labelDurasi2.Text = "/ " +
TimeSpan.FromSeconds(Math.Floor(durasi)).ToString();
                    //MENAMPILKAN NAMA FILE DAN NILAI
FPS DI PANEL OPENFILE
                    string namaFile =
openFile.FileName.Split('\\').Last();
                    //dk.insertTracking(namaFile);
                    //idTracking =
(int)dk.getMaxIdTracking()[0].Column1;
                    textBoxInput.Text =
openFile.FileName;
                    labelFileName.Text = namaFile;
                    labelFPS.Text =
frameRate.ToString() + " fps";
                    //MENAMPILKAN NAMA FILE, NILAI FPS,
DAN FRAME KE-*** DI TOOLSTRIPSTATUS
                    toolStripStatusLabel1.Text =
namaFile;
                    toolStripStatusLabel2.Text = "|  "
+ frameRate + " fps";
                    toolStripStatusLabel3.Text = "|  "
+ "ukuran " + baris + " x " + kolom;
                    toolStripStatusLabel5.Text =
namaFile;
```

LAMPIRAN A (LANJUTAN)

```
                    toolStripStatusLabel6.Text = "|  "
+ frameRate + " fps";
                    toolStripStatusLabel7.Text = "|  "
+ "ukuran " + baris + " x " + kolom;
                    //MENGAMBIL TIAP/NEXT FRAME DARI
OBJEK VIDEO
                    videoInput.ImageGrabbed += new
Emgu.CV.Capture.GrabEventHandler(prosesFrame);
                    //UPDATE TRACKBAR VIDEO MAIN
                    update_trackBarVideoMain(true);
//untuk mengaktifkan (enable/disable)
                    trackBarVideoMain.Minimum = 0;
                    trackBarVideoMain.Maximum =
(int)totalFrame;
                    trackBarVideoDetil.Minimum = 0;
                    trackBarVideoDetil.Maximum =
(int)totalFrame;
                }
                catch (NullReferenceException
exception)
                {
                    MessageBox.Show(exception.Message);
                }
                groupBoxPraProses.Enabled = true;
                groupBoxProses.Enabled = true;
            }
        }

        // EVENT KETIKA CHECKBOX ROI BERUBAH CENTANG
        private Rectangle userDefineRect;
        private Image<Bgr, byte> frameforSelectedRect;
        private void checkBox_ROI_CheckedChanged(object
sender, EventArgs e)
        {
            if (checkBox_ROI.Checked == true)
            {
                //Image<Bgr, byte> frame1st = new
Image<Bgr, byte>(videoInput.QueryFrame().ToBitmap());
```

LAMPIRAN A (LANJUTAN)

```
                //Bitmap frame_1st =
frame1st.ToBitmap();
                frameforSelectedRect = new Image<Bgr,
byte>(videoInput.QueryFrame().ToBitmap());
                Bitmap frame_1st =
frameforSelectedRect.ToBitmap();
                roiForm form = new roiForm();
                form.firstFrame = frame_1st;
                if (form.ShowDialog(this) ==
DialogResult.OK)
                {
                    pictureBoxMain.Image = form.bmCopy;
                    userDefineRect = form.rect;
                    textBox_RectX.Text =
userDefineRect.X.ToString();
                    textBox_RectY.Text =
userDefineRect.Y.ToString();
                    textBox_RectKol.Text =
userDefineRect.Width.ToString();
                    textBox_RectBrs.Text =
userDefineRect.Height.ToString();
                }
            }
            else
            {
                userDefineRect.X = 0;
                userDefineRect.Y = 0;
                userDefineRect.Width = 0;
                userDefineRect.Height = 0;
                textBox_RectX.Text = "0";
                textBox_RectY.Text = "0";
                textBox_RectKol.Text = "0";
                textBox_RectBrs.Text = "0";
            }
        }

        // EVENT KETIKA CHECKBOX GMM BERUBAH CENTANG --
> diambil semua
```

LAMPIRAN A (LANJUTAN)

```
        private deteksiGMM GMM = null;
        private void checkBox_GMM_CheckedChanged(object
sender, EventArgs e)
        {
            if (checkBox_GMM.Checked == true)
            {
                //textBox_TrainingFrame.Text =
Convert.ToString(30);
                textBox_KomponenGauss.Text =
Convert.ToString(4);
                textBox_LearningRate.Text =
Convert.ToString(0.001);
                comboBox_ShadowDetect.Text =
Convert.ToString(true);

                GMM = new deteksiGMM();
                GMM.inisialisasiGMM();
            }
            else
            {
                //textBox_TrainingFrame.Text = "";
                textBox_KomponenGauss.Text = "";
                textBox_LearningRate.Text = "";
                comboBox_ShadowDetect.Text = "";
            }
        }
        // EVENT KETIKA TEXTBOX TRAINING FRAME BERUBAH
        private void
textBox_TrainingFrame_TextChanged(object sender,
EventArgs e)
        {
            inisialisasiGMM();
        }
        // EVENT KETIKA TEXTBOX KOMPONEN GAUSSIAN
BERUBAH
        private void
textBox_KomponenGauss_TextChanged(object sender,
EventArgs e)
```

LAMPIRAN A (LANJUTAN)

```
        {
            inisialisasiGMM();
        }
        // EVENT KETIKA TEXTBOX LEARNING RATE BERUBAH
        private void
textBox_LearningRate_TextChanged(object sender,
EventArgs e)
        {
            inisialisasiGMM();
        }
        // EVENT KETIKA COMBOBOX SHADOW DETECT BERUBAH
        private void
comboBox_ShadowDetect_SelectedIndexChanged(object
sender, EventArgs e)
        {
            inisialisasiGMM();
        }
        // FUNGSI INISIALISASI GMM
        private void inisialisasiGMM()
        {
            if (textBox_KomponenGauss.Text != "" &&
textBox_LearningRate.Text != "" &&
comboBox_ShadowDetect.Text != "")
            {
                int komponenGauss =
Convert.ToInt32(textBox_KomponenGauss.Text);
                double learningRate =
Convert.ToDouble(textBox_LearningRate.Text);
                bool shadowDetect =
Convert.ToBoolean(comboBox_ShadowDetect.Text);
                GMM = new deteksiGMM(0, komponenGauss,
shadowDetect, learningRate);
                GMM.inisialisasiGMM();
            }
            else
                GMM = null;
        }
```

LAMPIRAN A (LANJUTAN)

```
        // EVENT KETIKA CHECKBOX FILTERING BERUBAH
CENTANG
        private applyFilter filter = null;
        private void
checkBox_Filtering_CheckedChanged(object sender,
EventArgs e)
        {
            if (checkBox_Filtering.Checked == true)
            {
                comboBox_Filtering.Text = "median";
                textBox_Kernel.Text =
Convert.ToString(9);

                filter = new applyFilter();
            }
            else
            {
                comboBox_Filtering.Text = "";
                textBox_Kernel.Text = "";
            }
        }
        // EVENT KETIKA comboBox_Filtering BERUBAH
        private void
comboBox_Filtering_SelectedIndexChanged(object sender,
EventArgs e)
        {
            inisialisasiFilter();
        }
        // EVENT KETIKA textBox_Kernel BERUBAH
        private void textBox_Kernel_TextChanged(object
sender, EventArgs e)
        {
            inisialisasiFilter();
        }
        // FUNGSI INISIALISASI Filter
        private void inisialisasiFilter()
        {
```

LAMPIRAN A (LANJUTAN)

```
            if (comboBox_Filtering.Text != "" &&
textBox_Kernel.Text != "")
            {
                string nmFilter =
comboBox_Filtering.Text;
                int kernel =
Convert.ToInt32(textBox_Kernel.Text);

                filter = new applyFilter(nmFilter,
kernel);
            }
            else
                filter = null;
        }

        // EVENT KETIKA CHECKBOX BLOB BERUBAH CENTANG3
        private deteksiBlob deteksiblob = null;
        private void
checkBox_Blob_CheckedChanged(object sender, EventArgs
e)
        {
            if (checkBox_Blob.Checked == true)
            {
                int minArea = 200;
                int maxArea = baris * kolom;
                int distance = 5;
                uint thInactive = 1;
                uint thActive = 2;

                textBox_MinArea.Text =
Convert.ToString(minArea);
                textBox_MaxArea.Text =
Convert.ToString(maxArea);
                textBox_thDistance.Text =
Convert.ToString(distance);
                textBox_thInactive.Text =
Convert.ToString(thInactive);
```

LAMPIRAN A (LANJUTAN)

```
                textBox_thActive.Text =
Convert.ToString(thActive);

                deteksiblob = new deteksiBlob(minArea,
maxArea, distance, thInactive, thActive);
            }
            else
            {
                textBox_MinArea.Text = "";
                textBox_MaxArea.Text = "";
                textBox_thDistance.Text = "";
                textBox_thActive.Text = "";
                textBox_thInactive.Text = "";
            }
        }
        // EVENT KETIKA textBox_MinArea BERUBAH
        private void textBox_MinArea_TextChanged(object
sender, EventArgs e)
        {
            inisialisasiBlob();
        }
        // EVENT KETIKA textBox_MaxArea BERUBAH
        private void textBox_MaxArea_TextChanged(object
sender, EventArgs e)
        {
            inisialisasiBlob();
        }
        // EVENT KETIKA textBox_thDistance BERUBAH
        private void
textBox_thDistance_TextChanged(object sender, EventArgs
e)
        {
            inisialisasiBlob();
        }
        // EVENT KETIKA textBox_thInactive BERUBAH
        private void
textBox_thInactive_TextChanged(object sender, EventArgs
e)
```

LAMPIRAN A (LANJUTAN)

```
        {
            inisialisasiBlob();
        }
        // EVENT KETIKA textBox_thActive BERUBAH
        private void
textBox_thActive_TextChanged(object sender, EventArgs
e)
        {
            inisialisasiBlob();
        }
        // FUNGSI INISIALISASI Blob
        private void inisialisasiBlob()
        {
            if (textBox_MinArea.Text != "" &&
textBox_MaxArea.Text != "" && textBox_thDistance.Text
!= "" && textBox_thInactive.Text != "" &&
textBox_thActive.Text != "")
            {
                int minArea =
Convert.ToInt32(textBox_MinArea.Text);
                int maxArea =
Convert.ToInt32(textBox_MaxArea.Text);
                int distance =
Convert.ToInt32(textBox_thDistance.Text);
                uint thInactive =
Convert.ToUInt32(textBox_thInactive.Text);
                uint thActive =
Convert.ToUInt32(textBox_thActive.Text);

                deteksiblob = new deteksiBlob(minArea,
maxArea, distance, thInactive, thActive);
            }
            else
                deteksiblob = null;
        }
```

LAMPIRAN A (LANJUTAN)

```
        // EVENT KETIKA BUTTON PLAY DI KLIK
        private bool statusVideoPLAY = false;
        private void buttonPlay_Click(object sender,
EventArgs e)
        {
            videoMulai();
        }
        // EVENT KETIKA BUTTON PAUSE DI KLIK
        private void buttonPause_Click(object sender,
EventArgs e)
        {
            videoPause();
        }
        // EVENT KETIKA BUTTON STOP DI KLIK
        private void buttonStop_Click(object sender,
EventArgs e)
        {
            videoStop();
        }
        // EVENT KETIKA BUTTON PLAY2 DI KLIK
        private void buttonPlay2_Click(object sender,
EventArgs e)
        {
            videoMulai();
        }
        // EVENT KETIKA BUTTON PAUSE2 DI KLIK
        private void buttonPause2_Click(object sender,
EventArgs e)
        {
            videoPause();
        }
        // EVENT KETIKA BUTTON STOP2 DI KLIK
        private void buttonStop2_Click(object sender,
EventArgs e)
        {
            videoStop();
```

LAMPIRAN A (LANJUTAN)

```
        }
        // FUNGSI START VIDEO
        private void videoMulai()
        {
            if (videoInput != null)
            {
                statusVideoPLAY = true;
                buttonPlay.Enabled = false;
                buttonPause.Enabled = true;
                buttonStop.Enabled = true;
                buttonPlay2.Enabled = false;
                buttonPause2.Enabled = true;
                buttonStop2.Enabled = true;
                if (statusVideoPLAY)
                {
                    //dk.deletKlasifikasi();
                    update_trackBarVideoMain(false);
                    videoInput.Start();
                }
            }
            else
            {
                //TAMPILKAN MESSAGE BOX BELUM ADA FILE
YANG TERPILIH
            }
        }
        // FUNGSI PAUSE VIDEO
        private void videoPause()
        {
            if (videoInput != null)
            {
                statusVideoPLAY = false;
                buttonPause.Enabled = false;
                buttonPlay.Enabled = true;
                buttonStop.Enabled = false;
                buttonPause2.Enabled = false;
                buttonPlay2.Enabled = true;
                buttonStop2.Enabled = false;
```

LAMPIRAN A (LANJUTAN)

```
                if (!statusVideoPLAY)
                {
                    videoInput.Pause();
                    update_trackBarVideoMain(true);
                }
            }
            else
            {
                //TAMPILKAN MESSAGE BOX BELUM ADA FILE
YANG TERPILIH
            }
        }
        // FUNGSI STOP VIDEO
        private void videoStop()
        {
            if (videoInput != null)
            {
                statusVideoPLAY = false;
                buttonPause.Enabled = false;
                buttonPlay.Enabled = true;
                buttonStop.Enabled = false;
                buttonPause2.Enabled = false;
                buttonPlay2.Enabled = true;
                buttonStop2.Enabled = false;

                if (!statusVideoPLAY)
                {
                    videoInput.Stop();
                    update_trackBarVideoMain(true);
                    framenumber = 0;

update_trackBarVideoMain(framenumber);

videoInput.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.C
V_CAP_PROP_POS_FRAMES, framenumber);
                    pictureBoxMain.Image = null;
                    prosesFrame(null, null);
                    resetCounting();
```

LAMPIRAN A (LANJUTAN)

```
                }
            }
            else
            {
                //TAMPILKAN MESSAGE BOX BELUM ADA FILE
YANG TERPILIH
            }
        }
        //RESET PENGHITUNGAN
        public void resetCounting()
        {
            ObjekList = new ArrayList();
            jumMobil = 0;
            jumMotor = 0;
            jumTruk = 0;
            jumUndefined = 0;
            idShow = 0;
            txtMotor.Text = jumMotor.ToString();
            txtMotor1.Text = jumMotor.ToString();
            txtMobil.Text = jumMobil.ToString();
            txtMobil1.Text = jumMobil.ToString();
            txtTruk.Text = jumTruk.ToString();
            txtTruk1.Text = jumTruk.ToString();
            txtUndefined.Text =
jumUndefined.ToString();
            txtUndefined1.Text =
jumUndefined.ToString();
            textBox_showJumlah1.Text =
idShow.ToString();
            textBox_showJumlah2.Text =
idShow.ToString();
        }
        // EVENT UNTUK PROSES TIAP FRAME KETIKA BUTTON
PLAY DI KLIK
        private Image<Bgr, Byte> frameAwal = null;
        private int framenumber;
        private Image<Gray, byte> foreground = null;
        private Image<Gray, byte> background = null;
```

LAMPIRAN A (LANJUTAN)

```
        private Image<Gray, Byte> citraTerfilter =
null;
        private Image<Gray, Byte> fgRemovShad = null;
        private Gray tresh = new Gray(254);
        private Image<Bgr, byte> frameROI = null;
        private Bgr warnaBiru = new Bgr(255, 0, 0);
        private Bgr warnaMerah = new Bgr(0, 0, 255);
        private uint iD;
        private uint idShow = 0, jumMotor=0,
jumMobil=0, jumTruk=0, jumUndefined=0;
        private Image<Gray, Byte> gray_image;
        private GeometricInvarianMoment GIM = new
GeometricInvarianMoment();
        private ArrayList ObjekList = new ArrayList();

        private double[,] GIMtarget;
        private string[] identitas;

        private MCvFont font = new
MCvFont(Emgu.CV.CvEnum.FONT.CV_FONT_HERSHEY_SIMPLEX,
0.5, 0.5);
        private int i = 1;
        void prosesFrame(object sender, EventArgs e)
        {

            frameAwal = new Image<Bgr,
byte>(videoInput.RetrieveBgrFrame().ToBitmap());
            // CEK KONDISI JIKA ROI DI SET OLEH USER
            if (userDefineRect.IsEmpty)
            {
                tampilVideo(frameAwal.ToBitmap(),
pictureBox1);
                frameROI = frameAwal.Clone();
            }
            else
            {
                tampilVideo(frameAwal.ToBitmap(),
pictureBox1);
```

LAMPIRAN A (LANJUTAN)

```
                frameROI =
frameAwal.Copy(userDefineRect);
                frameAwal.Draw(userDefineRect,
warnaBiru, 1);
            }
            // CEK KONDISI JIKA GMM DI SET OLEH USER
            if (checkBox_GMM.Checked == true)
            {
                GMM.prosesSubtraction(frameROI);
                foreground = GMM.foregroundMask;
                background = GMM.backgroundMask;
                tampilVideo(foreground.ToBitmap(),
pictureBox2);
                // CEK KONDISI JIKA FILTERING DI SET
OLEH USER
                if (checkBox_Filtering.Checked == true)
                {
                    filter.prosesFiltering(foreground);
                    citraTerfilter =
filter.filterImage;
 tampilVideo(citraTerfilter.ToBitmap(), pictureBox3);

                    fgRemovShad =
citraTerfilter.ThresholdToZero(tresh).Erode(3).Dilate(5
);
                    tampilVideo(fgRemovShad.ToBitmap(),
pictureBox4);
                }
                else
                {
                    fgRemovShad =
foreground.ThresholdToZero(tresh).Erode(4).Dilate(3);
                    citraTerfilter = null;
                    tampilVideo(null, pictureBox3);
                    tampilVideo(fgRemovShad.ToBitmap(),
pictureBox4);
                }
                if (framenumber < 125)
```

LAMPIRAN A (LANJUTAN)

```
                {
                    PointF koord = new PointF(130, 80);
                    frameAwal.Draw("Loading. . .", ref
font, Point.Round(koord), new Bgr(0.0, 0.0, 255.0));
                }
                // CEK KONDISI JIKA BLOB ANALYSIS DI
SET OLEH USER
                if (checkBox_Blob.Checked == true &&
framenumber>=125)
                {
                    deteksiblob.proses(fgRemovShad);

                    if (deteksiblob.blobs.Count != 0)
                    {
                        deteksiblob.tracking();
                        frameAwal.Draw(userDefineRect,
warnaMerah, 1);
                    }

                    //-------- Cek Centroid-Begin -----
----
                    string id = null;
                    string gim = null;
                    gray_image = null;
                    //-------- Proses Klasifikasi dan
Penghitungan Kendaraan - Begin ---------
                    foreach (KeyValuePair<UInt32,
CvTrack> tr in deteksiblob.tracks)
                    {
                        gray_image =
frameROI.Convert<Gray,
Byte>().Copy(tr.Value.BoundingBox);

                        GIM = new
GeometricInvarianMoment(gray_image);

                        PointF koord = new
PointF((float)tr.Value.BoundingBox.X +
```

LAMPIRAN A (LANJUTAN)

```
userDefineRect.X, (float)tr.Value.BoundingBox.Y +
userDefineRect.Y);

frameAwal.Draw(GIM.Recognizing(), ref font,
Point.Round(koord), new Bgr(255.0, 0.0, 0.0));
                    if (tr.Value.BoundingBox.Y != 0
&& tr.Value.BoundingBox.Y != userDefineRect.Height &&
tr.Value.Lifetime >6 &&
!isExistIdObjek(tr.Value.Id.ToString()))
                    {
ObjekList.Add(tr.Value.Id.ToString());

updateCounting(GIM.Recognizing());
                                idShow = jumMobil +
jumMotor + jumTruk + jumUndefined;
                            //}
                    }
                }
                //-------- Proses Klasifikasi dan
Penghitungan Kendaraan - End ---------

                foreach (KeyValuePair<UInt32,
CvTrack> tr in deteksiblob.tracks)
                {
                    //iD = tr.Value.Id;
                    //if (idShow < iD)
                    //{
                    //    idShow = iD;
                    //}
                    PointF koord = new
PointF((float)tr.Value.BoundingBox.X +
userDefineRect.X, (float)tr.Value.BoundingBox.Y +
userDefineRect.Y);
                    Rectangle kotak = new
Rectangle();
                    kotak = tr.Value.BoundingBox;
                    kotak.X = kotak.X +
userDefineRect.X;
```

LAMPIRAN A (LANJUTAN)

```
                    kotak.Y = kotak.Y +
userDefineRect.Y;
                    frameAwal.Draw(kotak, new
Bgr(0, 255, 0), 1);
                }
                //-------- Memberi Kotak pada
objek-End ---------
                textBox_showJumlah1.Text =
idShow.ToString();
                textBox_showJumlah2.Text =
idShow.ToString();
            }
            tampilVideo(frameAwal.ToBitmap(),
pictureBox5);
        }
        else
        {
            tampilVideo(frameAwal.ToBitmap(),
pictureBox5);
            foreground = null;
            citraTerfilter = null;
            fgRemovShad = null;
            tampilVideo(null, pictureBox2);
            tampilVideo(null, pictureBox3);
            tampilVideo(null, pictureBox4);
        }
        tampilVideo(frameAwal.ToBitmap(),
pictureBoxMain);
        //UPDATE WAKTU TAYANG VIDEO
        double time_index =
videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.C
V_CAP_PROP_POS_MSEC);
        labelTime.Text =
TimeSpan.FromSeconds(Math.Ceiling(time_index /
1000.0)).ToString();
        labelTime2.Text =
TimeSpan.FromSeconds(Math.Ceiling(time_index /
1000.0)).ToString();
```

LAMPIRAN A (LANJUTAN)

```
            //SHOW FRAME NUMBER
            framenumber =
(int)videoInput.GetCaptureProperty(Emgu.CV.CvEnum.CAP_P
ROP.CV_CAP_PROP_POS_FRAMES);
            toolStripStatusLabel4.Text = "|   " + "Frame
ke- " + framenumber.ToString();
            toolStripStatusLabel8.Text = "|   " + "Frame
ke- " + framenumber.ToString();
            //UPDATE TRACKBAR
            update_trackBarVideoMain(framenumber);
            //SATUAN WAKTU (DELAY) SHOW FRAME DI
PICTURE BOX
            Thread.Sleep((int)(1000.0 / frameRate));

            if (framenumber == totalFrame)
            {
                videoInput.Stop();
            i++;
        }
        //UPDATE PENGHITUNGAN KENDARAAN SESUAI JENIS
        public void updateCounting(string jenis)
        {
            if (jenis == "motor")
            {
                jumMotor++;
                txtMotor.Text = jumMotor.ToString();
                txtMotor1.Text = jumMotor.ToString();
            }
            else if (jenis == "mobil")
            {
                jumMobil++;
                txtMobil.Text = jumMobil.ToString();
                txtMobil1.Text = jumMobil.ToString();
            }
            else if (jenis == "truk")
            {
                jumTruk++;
                txtTruk.Text = jumTruk.ToString();
```

LAMPIRAN A (LANJUTAN)

```
                txtTruk1.Text = jumTruk.ToString();
            }
            else if (jenis == "")
            {
                jumUndefined++;
                txtUndefined.Text =
jumUndefined.ToString();
                txtUndefined1.Text =
jumUndefined.ToString();
            }
        }
        //CEK APAKAH ID OBJEK TELAH DIHITUNG?
        public bool isExistIdObjek(string id)
        {
            bool exist=false;
            foreach (string objek in ObjekList)
            {
                if (objek == id)
                    exist = true;
            }
            return exist;
        }

        // UNTUK MENAMPILKAN HASIL VIDEO KE INTERFACE
        private void tampilVideo(Bitmap frame,
PictureBox pictureBoxDisplay)
        {
            pictureBoxDisplay.SizeMode =
PictureBoxSizeMode.Zoom;
            if (frame == null)
                pictureBoxDisplay.Image = null;
            else
                pictureBoxDisplay.Image = frame;
        }

        //UNTUK UPDATE TRACKBAR VALUE
        private delegate void
update_trackBarVideoMainDelegate(int Value);
```

LAMPIRAN A (LANJUTAN)

```
        private void update_trackBarVideoMain(int
Value)
        {
            if (trackBarVideoMain.InvokeRequired)
            {
                try
                {
                    update_trackBarVideoMainDelegate
UVC = new
update_trackBarVideoMainDelegate(update_trackBarVideoMa
in);
                    this.BeginInvoke(UVC, new object[]
{ Value });
                }
                catch (Exception ex)
                {
                }

            }
            else
            {
                if (Value < trackBarVideoMain.Maximum)
                {
                    trackBarVideoMain.Value = Value;
                    trackBarVideoDetil.Value = Value;
                }

            }
        }

        //UNTUK UPDATE TRACKBAR ENABLE/DISABLE
        private delegate void
EnabletrackBarVideoMainDelegate(bool State);
        private void update_trackBarVideoMain(bool
State)
        {
            if (trackBarVideoMain.InvokeRequired)
            {
```

LAMPIRAN A (LANJUTAN)

```
                try
                {
                    EnabletrackBarVideoMainDelegate UVC
= new
EnabletrackBarVideoMainDelegate(update_trackBarVideoMai
n);
                    this.BeginInvoke(UVC, new object[]
{ State });
                }
                catch (Exception ex)
                {
                }
            }
            else
            {
                //Do a quick in range check as sometime
the codec may not tell the truth
                trackBarVideoMain.Enabled = State;
                trackBarVideoDetil.Enabled = State;
            }
        }

        // EVENT KETIKA POSISI TRACKBAR BERUBAH
        private void
trackBarVideoMain_MouseCaptureChanged(object sender,
EventArgs e)
        {
            if (videoInput != null)
            {
                //we don't use this when running since
it has an unstable call and wil cause a crash
                if (videoInput.GrabProcessState ==
System.Threading.ThreadState.Running)
                {
                    videoInput.Pause();
                    while (videoInput.GrabProcessState
== System.Threading.ThreadState.Running) ;//do nothing
wait for stop
```

LAMPIRAN A (LANJUTAN)

```
videoInput.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.C
V_CAP_PROP_POS_FRAMES, trackBarVideoMain.Value);
                    videoInput.Start();
                }
                else
                {

videoInput.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.C
V_CAP_PROP_POS_FRAMES, trackBarVideoMain.Value);
                    //call the process frame to update
the picturebox
                    prosesFrame(null, null);

                }
            }
        }
        // EVENT KETIKA POSISI TRACKBAR DETIL BERUBAH
        private void
trackBarVideoDetil_MouseCaptureChanged(object sender,
EventArgs e)
        {
            if (videoInput != null)
            {
                //we don't use this when running since
it has an unstable call and wil cause a crash
                if (videoInput.GrabProcessState ==
System.Threading.ThreadState.Running)
                {
                    videoInput.Pause();
                    while (videoInput.GrabProcessState
== System.Threading.ThreadState.Running) ;//do nothing
wait for stop

videoInput.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.C
V_CAP_PROP_POS_FRAMES, trackBarVideoDetil.Value);
                    videoInput.Start();
                }
```

LAMPIRAN A (LANJUTAN)

```
                else
                {
videoInput.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.C
V_CAP_PROP_POS_FRAMES, trackBarVideoDetil.Value);
                    //call the process frame to update
the picturebox
                    prosesFrame(null, null);
                }
            }
        }


        private Rectangle minArea;
        private void
linkLabel_MinArea_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            //Image<Bgr, byte> frame1st = new
Image<Bgr, byte>(videoInput.QueryFrame().ToBitmap());
            Bitmap frame_1st =
frameforSelectedRect.ToBitmap();
            roiForm form = new roiForm();
            form.firstFrame = frame_1st;
            if (form.ShowDialog(this) ==
DialogResult.OK)
            {
                minArea = form.rect;
                textBox_MinArea.Text =
Convert.ToString(minArea.Width * minArea.Height);
            }

        }

        private Rectangle maxArea;
        private void
linkLabel_MaxArea_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
```

LAMPIRAN A (LANJUTAN)

```
        {
            //Image<Bgr, byte> frame1st = new
Image<Bgr, byte>(videoInput.QueryFrame().ToBitmap());
            Bitmap frame_1st =
frameforSelectedRect.ToBitmap();
            roiForm form = new roiForm();
            form.firstFrame = frame_1st;
            if (form.ShowDialog(this) ==
DialogResult.OK)
            {
                maxArea = form.rect;
                textBox_MaxArea.Text =
Convert.ToString(maxArea.Width * maxArea.Height);
            }

        }

        protected override void
OnClosing(CancelEventArgs e)
        {
            if (videoInput != null)
            {
                if (videoInput.GrabProcessState ==
System.Threading.ThreadState.Running)
                    videoInput.Stop();
                videoInput.Dispose();
            }
        }

    }
}
```

LAMPIRAN A (LANJUTAN)

A.2 Kode Program Class roiForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace VehicleCounting
{
    public partial class roiForm : Form
    {

        public roiForm()
        {
            InitializeComponent();
        }


        public Bitmap firstFrame
        {
            set { pictureBox_ROI.Image = value; }
        }


        // Position of selected rectangel from user
        private Point startRect;
        private Size dimensiRect;

        // VARIABEL RECTANGLE (MENGAMBIL POINT YANG
DIPILIH USER)
        public Rectangle rect;

        // The original image.
        private Bitmap bmOriginal;
```

LAMPIRAN A (LANJUTAN)

```
        // The copy image.
        public Bitmap bmCopy;

        // True when we will select a rectangle
        private bool IsBtnRectSelected = false;

        // True when we're selecting a rectangle.
        private bool IsSelecting = false;

        // Point start (x0,y0) and point selected
(x1,y1)
        private int x0, y0, x1, y1;


        // On first displaying of the form
        protected override void OnLoad(EventArgs e)
        {
            // get video frame dimension and set new
form's size
            if (pictureBox_ROI.Image != null)
            {
                int imageWidth =
pictureBox_ROI.Image.Width;
                int imageHeight =
pictureBox_ROI.Image.Height;

                // resize region definition control
                pictureBox_ROI.Size = new
Size(imageWidth, imageHeight);
                // resize window
                this.Size = new Size(imageWidth + 80,
imageHeight + 110);

                bmOriginal = new
Bitmap(pictureBox_ROI.Image);
            }
```

LAMPIRAN A (LANJUTAN)

```
            base.OnLoad(e);
        }

        private void pictureBox_ROI_MouseDown(object
sender, MouseEventArgs e)
        {
            if (IsBtnRectSelected)
            {
                IsSelecting = true;

                // Save the start point.
                x0 = e.X;
                y0 = e.Y;
            }
        }

        private void pictureBox_ROI_MouseMove(object
sender, MouseEventArgs e)
        {
            if (IsBtnRectSelected)
            {
                // Do nothing it we're not selectign an
area.
                if (!IsSelecting) return;

                // Save the new point.
                x1 = e.X;
                y1 = e.Y;

                // Make a Bitmap to display the
selection rectangle.
                bmCopy = new Bitmap(bmOriginal);


                startRect = new Point(Math.Min(x0, x1),
Math.Min(y0, y1));
                dimensiRect = new Size(Math.Abs(x0 -
x1), Math.Abs(y0 - y1));
```

LAMPIRAN A (LANJUTAN)

```
                rect = new Rectangle(startRect,
dimensiRect);

                // Draw the rectangle.
                using (Graphics gr =
Graphics.FromImage(bmCopy))
                {
                    gr.DrawRectangle(Pens.Red,rect);
                }

                // Display the temporary bitmap.
                pictureBox_ROI.Image = bmCopy;
            }
        }

        private void pictureBox_ROI_MouseUp(object
sender, MouseEventArgs e)
        {
            if (IsBtnRectSelected)
            {
                // Do nothing it we're not selecting an
area.
                if (!IsSelecting) return;
                IsSelecting = false;

                // Draw the rectangle.
                using (Graphics gr =
Graphics.FromImage(bmCopy))
                {
                    gr.DrawRectangle(Pens.Blue, rect);
                }

                pictureBox_ROI.Image = bmCopy;

                // Display the result.
                //picResult.Image = area;
                rectangleButton.Checked = false;
```

LAMPIRAN A (LANJUTAN)

```
                IsBtnRectSelected = false;
            }
        }

        private void rectangleButton_Click(object
sender, EventArgs e)
        {
            rectangleButton.Checked = true;
            IsBtnRectSelected = true;
            pictureBox_ROI.Cursor =
System.Windows.Forms.Cursors.Cross;
        }


    }
}
```

A.3 Kode Program Class deteksiGMM.cs[5]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Drawing;

using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Emgu.CV.VideoSurveillance;
using Emgu.CV.Cvb;

namespace VehicleCounting
{
    class deteksiGMM
    {
        // Field
```

LAMPIRAN A (LANJUTAN)

```
        private BackgroundSubstractorMOG2 GMM;
        private int numOfTrainFrame = 0;
        private int komponenGauss;
        private bool shadowDetect;
        private double learningRate;
        private Image<Gray, byte> foregroundImage;

        // Property
        public Image<Gray, byte> foregroundMask
        {
            get { return foregroundImage; }
        }

        // Constructor
        public deteksiGMM()
        {
            this.numOfTrainFrame = 0;
            this.komponenGauss = 4;
            this.shadowDetect = true;
            this.learningRate = 0.001;
        }
        // Overloading Constructor
        public deteksiGMM(int trainFrame, int nGauss,
bool detectShadow, double lrate)
        {
            this.numOfTrainFrame = trainFrame;
            this.komponenGauss = nGauss;
            this.shadowDetect = detectShadow;
            this.learningRate = lrate;
        }

        // Destructor
        ~deteksiGMM()
        {
            GMM = null;
            foregroundImage = null;
        }
```

```
        // Method
        public void inisialisasiGMM()
        {
            GMM = new
BackgroundSubstractorMOG2(numOfTrainFrame,
komponenGauss, shadowDetect);
        }
        public void prosesSubtraction(Image<Bgr,byte>
frameInput)
        {
            GMM.Update(frameInput, this.learningRate);
            this.foregroundImage = GMM.ForgroundMask;
        }
    }
}
```

A.4 Kode Program Class applyFilter.cs[5]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;

using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Emgu.CV.Cvb;

namespace VehicleCounting
{
    class applyFilter
    {
        // Field
        private string jenisFilter = null;
        private int ukuranKernel;
        private Image<Gray, byte> frameFilter;
```

LAMPIRAN A (LANJUTAN)

```
        // Property
        public Image<Gray, byte> filterImage
        {
            get { return frameFilter; }
        }

        // Constructor
        public applyFilter()
        {
            this.jenisFilter = "median";
            this.ukuranKernel = 9;
        }
        // Overloading Constructor
        public applyFilter(string jnsFilter, int
sizeKernel)
        {
            this.jenisFilter = jnsFilter;
            this.ukuranKernel = sizeKernel;
        }

        // Destructor
        ~applyFilter()
        {
            frameFilter = null;
        }

        // Method
        public void prosesFiltering(Image<Gray,byte>
frameInput)
        {
            if(jenisFilter.Equals("median"))
            {
                frameFilter =
frameInput.SmoothMedian(ukuranKernel);
            }
            else if (jenisFilter.Equals("gaussian"))
            {
```

LAMPIRAN A (LANJUTAN)

```
                frameFilter =
frameInput.SmoothGaussian(ukuranKernel, ukuranKernel,
0.1, 0.1);
            }
            else if (jenisFilter.Equals("blur"))
            {
                frameFilter =
frameInput.SmoothBlur(ukuranKernel, ukuranKernel,
true);
            }
            else if (jenisFilter.Equals("bilatral"))
            {
                frameFilter =
frameInput.SmoothBilatral(ukuranKernel, 1, 1);
            }
        }
    }
}
```

A.5 Kode Program Class deteksiBlob.cs[5]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;

using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Emgu.CV.Cvb;

namespace VehicleCounting
{
    class deteksiBlob
    {
```

LAMPIRAN A (LANJUTAN)

```
// Field
private Image<Gray, byte> fgImage;
private List<PointF> centroidsDetected;
private List<Rectangle> bboxesDetected;
private List<int> areasDetected;
private int minArea;
private int maxArea;
private int distanceBlob;
private uint thInactive;
private uint thActive;
private CvBlobDetector blobDetect;
private CvBlobs blobsDetected;
private CvTracks track;


// Property
public List<PointF> centroids
{
    get { return centroidsDetected; }
    set { centroidsDetected = value; }
}
public List<Rectangle> bboxes
{
    get { return bboxesDetected; }
    set { bboxesDetected = value; }
}
public List<int> areas
{
    get { return areasDetected; }
    set { areasDetected = value; }
}
public CvBlobs blobs
{
    get { return blobsDetected; }
    set { blobsDetected = value; }
}
public CvTracks tracks
{
```

LAMPIRAN A (LANJUTAN)

```
            get { return track; }
            set { track = value; }
        }

        // Constructor
        public deteksiBlob(int minimalArea, int
maksimalArea, int distance, uint InAktif, uint Aktif)
        {
            this.minArea = minimalArea;
            this.maxArea = maksimalArea;
            this.distanceBlob = distance;
            this.thInactive = InAktif;
            this.thActive = Aktif;
            track = new CvTracks();
        }

        // Destructor
        ~deteksiBlob()
        {
            fgImage = null;
            centroidsDetected = null;
            bboxesDetected = null;
            areasDetected = null;
            blobDetect = null;
            blobsDetected = null;
            track = null;
        }


        // Methods
        public void proses(Image<Gray, Byte>
foregroundImage)
        {
            this.fgImage = foregroundImage.Clone();
            bboxesDetected = new List<Rectangle>();
            centroidsDetected = new List<PointF>();
            areasDetected = new List<int>();
```

LAMPIRAN A (LANJUTAN)

```
            blobDetect = new
Emgu.CV.Cvb.CvBlobDetector();
            blobsDetected = new CvBlobs();

            blobDetect.Detect(fgImage, blobsDetected);
            blobsDetected.FilterByArea(minArea,
maxArea);

            foreach (KeyValuePair<uint, CvBlob> blob in
blobsDetected)
            {

centroidsDetected.Add(blob.Value.Centroid);

bboxesDetected.Add(blob.Value.BoundingBox);
                areasDetected.Add(blob.Value.Area);
            }
        }

        public void tracking()
        {
            track.Update(this.blobsDetected,
this.distanceBlob, this.thInactive, this.thActive);
        }

    }
}
```

LAMPIRAN A (LANJUTAN)

A.6 Kode Program Class GeometricInvariantMoment.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;

namespace VehicleCounting
{
    class GeometricInvarianMoment
    {
        //Field
        private Image<Gray, Byte> gray_image;
        private MCvMoments Moments;
        private MCvHuMoments HuMoments;
        private double[] HuM = new double[7];
        int id = 8, idmin, area;
        double sigma;
        double[] d = new double[100];
        double[,] GIMtarget;
        string[] identitas;

        //konstruktor
        public GeometricInvarianMoment() {}
        public GeometricInvarianMoment(Image<Gray,
Byte> grayImage)
        {
            this.gray_image = grayImage;

            this.Moments = grayImage.GetMoments(true);
            this.HuMoments = Moments.GetHuMoment();
            this.setHuMoments(HuMoments);
        }
```

LAMPIRAN A (LANJUTAN)

```
        //destruktor
        ~GeometricInvarianMoment() { }

        // Methods
        public void setHuMoments(MCvHuMoments Hu)
        {
            this.HuM[0] = Hu.hu1;
            this.HuM[1] = Hu.hu2;
            this.HuM[2] = Hu.hu3;
            this.HuM[3] = Hu.hu4;
            this.HuM[4] = Hu.hu5;
            this.HuM[5] = Hu.hu6;
            this.HuM[6] = Hu.hu7;
        }
        public double[] getHuMoments()
        {
            return this.HuM;
        }

        public string Recognizing()
        {
            if (this.getHuMoments()[0] >= 0.174 &&
this.getHuMoments()[0] <= 1)
            {
                if (this.gray_image.Width <= 30)
                {
                    return "motor";
                }
                else if (this.gray_image.Width > 30 &&
this.gray_image.Width <= 92)
                {
                    return "mobil";
                }
                else
                {
                    return "truk";
                }
            }
```

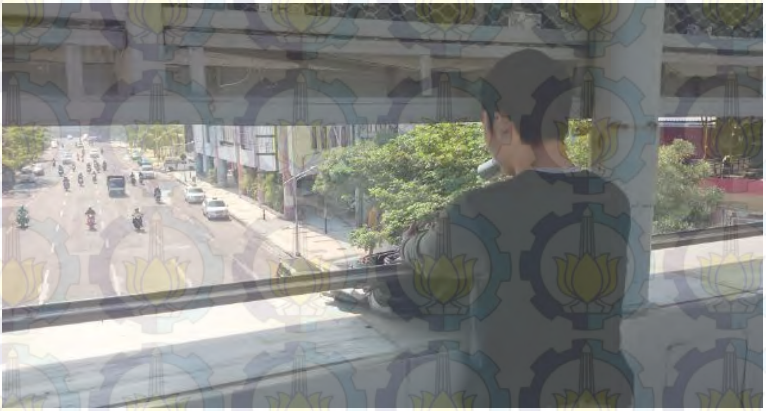
LAMPIRAN A (LANJUTAN)

```
            else if (this.getHuMoments()[0] >= 0.16 &&
this.getHuMoments()[0] < 0.174)
            {
                if (this.gray_image.Width <= 30)
                {
                    return "motor";
                }
                else if (this.gray_image.Width > 30 &&
this.gray_image.Width <= 92)
                {
                    return "mobil";
                }
                else
                {
                    return "truk";
                }
            }
            else
            {
                return "";
            }
        }
    }
}
```

LAMPIRAN B

B.1 Dokumentasi Pengambilan Video di Jalan Embong Malang

Gambar B.1.1 Jembatan Penyeberangan di Jalan Embong Malang



Gambar B.1.1 Posisi Pengambilan Video diatas Jembatan Penyeberangan

LAMPIRAN B (LANJUTAN)

B.2 Hasil Uji Coba Klasifikasi Jenis Kendaraan di Jalan Embong Malang

Gambar B.2.1 Antar Muka Hasil Uji Coba Klasifikasi Jenis Kendaraan di Jalan Embong Malang Pagi Hari

Gambar B.2.2 Antar Muka Hasil Uji Coba Klasifikasi Jenis Kendaraan di Jalan Embong Malang Siang Hari

DAFTAR LAMPIRAN

"Halaman ini sengaja dikosongkan"